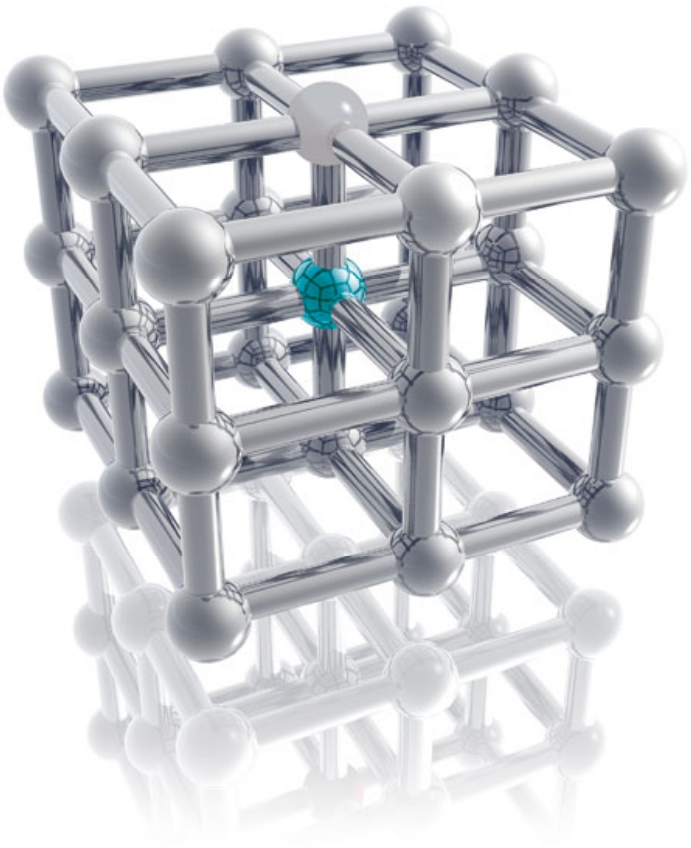


asix – User's Manual

Doc. No. ENP5062
Version: 28-05-2008

ASKOM® and **asix** ® are registered trademarks of ASKOM Spółka z o.o., Gliwice. Other brand names, trademarks, and registered trademarks are the property of their respective holders.

All rights reserved including the right of reproduction in whole or in part in any form. No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, without prior written permission from the ASKOM.

ASKOM sp. z o. o. shall not be liable for any damages arising out of the use of information included in the publication content.

Copyright © 2008, ASKOM Sp. z o. o., Gliwice

Table of Contents

1. WHAT`S NEW IN ASIX5 7

2. HELP SYSTEM 11

3. INSTALLING AND CONFIGURING THE ASIX SOFTWARE 15

3.1. Hardware & Software Requirements 15
3.2. Installing asix 15
3.3. Starting asix 22
3.4. Uninstallation 23

4. STAGES OF APPLICATION DEVELOPMENT 25

4.1. Defining the Process Data for ASMEN Program 26
4.2. Defining the Process Data to Be Archived 26
4.3. Creating an XML - application configuration file 27
4.4. Creating a Variable Definitions Database 27
4.5. Creating the Visual Masks 28
4.6. Selecting the Application Handling Mode 28
4.7. Defining the Application Operating Parameters 29
4.8. Running the Application 30
4.9. Network Installations 31

5. SELECTION OF THE ASIX SYSTEM LANGUAGE 33

6. COMMUNICATION MANAGER ASMEN 37

6.1. Declaration of Transmission Channels 39
6.2. Declaration of Process Variables 42
6.3. Running the ASMEN Module 43
6.4. Entries in Configuration File 43
6.5. Conversion Functions 44
6.6. Diagnostics of Correct Format of Numbers Read from Controllers 69
6.7. Drivers and Transmission Protocols 69
6.8. Data Exchange in Local Area Network 73
6.8.1. Declaration of Network Channel 73
6.8.2. Process Variable Declarations in Network Protocol 76
6.8.3. GATEWAY Station Operating-Mode 76
6.8.4. Redundancy 77

7. ASPAD - DATA ARCHIVING MODULE 81

7.1.1. Basic Features of ASPAD Module 81
7.1.2. Basic Kinds and Types of ASPAD Archives 82
7.1.3. Configuration parameters of Aspad module 83
7.2. ASPAD Modules 84
7.3. System Requirements 84
7.3.1. RAM Memory 84
7.3.2. Disk Memory for the D, M, Y and H Archives 85
7.3.3. Disk Memory for the B Archive 86
7.4. The Files Created by Programs of the ASPAD Module 86
7.4.1. Archive Files of D, M, Y and H-Type 87
7.4.2. Archive Files of B and P-Type 88
7.5. Configuring the System During the ASPAD Module Installation 89
7.6. Configuring and Starting the ASPAD Module 89
7.6.1. Specification of Time Periods in Configuration Files 90
7.6.2. Definition of the Basic Parameters of Data Collecting 90
7.6.3. Automatic Backup 91
7.6.4. Automatic Deleting of the Oldest Files 91
7.6.5. Access to ASPAD Data via Network 91
7.6.6. Variable Archiving Parameters 93
7.6.7. Atributes of Archiving 95
7.7. Conditional Archiving 97
7.7.1. Declaration of Conditions 97

7.7.2. Archiving Condition Operators ... 98
7.7.3. Condition Examples ... 99
7.8. B-TYPE DATA COLLECTION ... 100
7.8.1. Special Features of B-Type Data Collection ... 100
7.8.2. Additional Requirements for B-Type Archives ... 100
7.8.3. Replenishment of Data in the B-Type Archives ... 102
7.8.4. Synchronization with a Dual Database ... 103
7.8.5. Database Declaration ... 104
7.8.6. Declaring the Variables for Archiving in Database ... 104
7.8.7. Additional Information about Archived Points ... 104
7.8.8. Archiving in Multiple Databases ... 106
7.8.9. Insequential Archiving ... 106
7.8.10. Automatic Creation of the Archive Directory and Database Name in Paradox Environment ... 106
7.8.11. Browsing the Contents of B-type Archive Database with the BBrowse Browser ... 106
7.9. PATTERN TRENDS ... 107
7.9.1. Pattern Trends Declaration ... 107
7.9.2. Generation of Pattern Trends ... 107

8. ASLINK - NETWORK MODULE ... 109

8.1. GENERAL CHARACTERISTICS OF THE ASLINK MODULE ... 109
8.1.1. Services of Reliable Data Transmission ... 109
8.1.2. Time Synchronization Service ... 109
8.1.3. Remote File Access Services ... 110
8.1.4. Multi-Board (PCB) Configurations ... 110
8.2. HARDWARE AND SOFTWARE REQUIREMENTS ... 110
8.2.1. Types of Supported Local Area Networks (LANs) ... 110
8.2.2. Software Requirements ... 111
8.2.3. Remote Access Service (RAS) Activation in Windows XP/2003 ... 111
8.3. CONFIGURATION OF ASLINK MODULE ... 112
8.3.1. Station Identification ... 112
8.3.2. Access to the Network ... 112
8.3.3. Time Synchronization ... 114
8.3.4. Protection with Password ... 115
8.3.5. Connection to Previous ASLINK Version Workstations ... 116
8.3.6. Diagnostics ... 116

9. DESIGNER – DESIGNING THE APPLICATION ... 117

9.1. DESIGNER WINDOW ... 117
9.1.1. Components of Designer Window ... 117
9.1.2. Description of Designer Window ... 119
9.2. CREATION OF SYNOPTIC MASKS ... 124
9.2.1. Mask Definition Window ... 124
9.2.2. Mask Selection Window ... 127
9.2.3. Description of MASKS Menu Commands ... 128
9.2.4. Context-Sensitive Menu for Mask Edition ... 129
9.2.5. Object Edition ... 129
9.2.6. Replacement of Process Variables During Mask Opening ... 139
9.3. COMPOSING MASKS ... 140
9.3.1. Key Identification Rules ... 140
9.3.2. Defining Keyboard Shortcuts ... 141
9.4. CONTROL OPERATIONS ... 144
9.4.1. Operator Actions ... 145
9.5. PASSWORDS ... 146
9.5.1. Protection of Control Operations ... 147
9.5.2. Protection of Action Execution ... 148
9.5.3. Operator Actions ... 149
9.6. LOGON SYSTEM ... 149
9.7. SYSTEM PROTECTIONS ... 152
9.8. USING BITMAPS ... 154
9.8.1. Bitmap Editor ... 156
9.9. USING EXPRESSIONS ... 161
9.10. USING VARIABLES ... 163

9.11. Fonts Usage 164
9.11.1. Directions for Polish Character Use 167
9.12. The List of Key Operations 168
9.12.1. Macros 170

10. EXECUTIVE - RUNNING THE APPLICATION 173

10.1. Control Panel 173
10.2. Components of the Control Panel 174
10.2.1. Description of the Control Panel 175
10.2.2. Switching Over the Visual Masks 176
10.2.3. Operator's Actions 177
10.3. Using the Printer 177
10.3.1. Dump of Screen Contents 179
10.4. Time Setting 181
10.4.1. Operator's Actions 182
10.5. Demonstration Sequences 182
10.6. Screen Keyboard Mechanism 183

11. VARIABLE TABLES 185

11.1. Opening the Table 185
11.2. Structure of Table Definition File 185
11.3. User Interface 191
11.4. Binary Signals 195

12. ALARM SYSTEM 197

12.1. Configuration of Alarm Manager 198
12.1.1. Files of Alarm Definitions 198
12.1.2. Files of Group Definitions 200
12.1.3. Control of Cumulative Alarm Group State 200
12.1.4. Selection of Alarms Printed in the On-line Mode 201
12.1.5. Strategies of Alarm Detections 201
12.1.6. Configuring Network Operation 210
12.1.7. Exclusions and Filters 212
12.1.8. Sound Signals 217
12.1.9. Control Log 217
12.1.10. Operator Actions 218
12.2. Alarm Masks 219
12.2.1. Handling of Alarm Masks 219
12.2.2. Defining Alarm Masks 231

13. REPORT GENERATION MODULE 235

13.1. Handling Reports 235
13.1.1. Printing Reports 243
13.1.2. Handling Dialogs 244
13.1.3. Operator Actions 244
13.1.4. Report System Handling 244
13.2. Designing Reports 244
13.2.1. The Simplest Report 245
13.2.2. Files 246
13.2.3. Description of Fields 246
13.2.4. Definition of Reports 247
13.2.5. The Examplary Report 259
13.3. ASTEL - the Language for Computing the Process Data 261
13.3.1. Components of the Language 262
13.3.2. Grammar of the ASTEL Language 275
13.3.3. Evaluation of Expressions 277
13.3.4. Diagnostics 279
13.4. ASTER - the Language for Report Definition 285
13.4.1. Components of the Language 286
13.4.2. Instructions 291
13.4.3. The ASTER Language Grammar 300
13.4.4. Diagnostics 301
13.5. Script Reports (VBScript, JavaScript) 305

13.5.1. Declaring Script Reports ... 305
13.5.2. Raporter Window ... 305
13.5.3. Operator Actions ... 305
13.5.4. Rules of Creating Script Reports ... 306
13.5.5. Meaning of Script Parameters ... 306
13.5.6. Printout Control ... 307
13.5.7. Codepage of Object Files ... 308

14. MULTI-MONITOR SYSTEMS ... 309

15. AUTHORIZATION CONTROL SYSTEM ... 311

16. OPERATOR ACTIONS ... 313

16.1. Operator Actions Reference List ... 313
16.2. Description of Actions ... 315

17. ACTIONS SCHEDULER ... 335

17.1. Definition of Actions Schedules ... 335
17.1.1. Defining the Time-Depending Actions ... 336
17.1.2. Defining the Event-Depending Actions ... 337
17.1.3. Storage of Parameters ... 338
17.1.4. Execution of Time-Depending Actions ... 338

18. OBJECTS OF THE ASIX SYSTEM ... 341

18.1. Set of Objects of the asix System ... 341
18.2. Glossary ... 343
18.3. Blinking Attribute ... 344
18.3.1. Blinking Declaration ... 345
18.3.2. Setting Blinking Frequency ... 346
18.4. Objects ... 347
18.4.1. ASBASE Object ... 347
18.4.2. BAR Object ... 352
18.4.3. BUTTON Object ... 357
18.4.4. CALCULATOR Object ... 361
18.4.5. DATE+TIME Object ... 366
18.4.6. ELLIPSE Object ... 369
18.4.7. ELLIPSES Object ... 371
18.4.8. EXPESSION Object ... 377
18.4.9. LINE and LINE HV Objects ... 383
18.4.10. LINES and LINES HV Objects ... 386
18.4.11. MESSAGES Object ... 392
18.4.12. MOVE CONTROLLER Object ... 397
18.4.13. MOTOR Object ... 398
18.4.14. NUMBER Object ... 402
18.4.15. OBJECTS CONTROLLER Object ... 410
18.4.16. STRING Object ... 411
18.4.17. PICTURE Object ... 415
18.4.18. PICTURES Object ... 416
18.4.19. PIPELINE Object ... 423
18.4.20. POINTER Object ... 428
18.4.21. POLYGON and POLYGON HV Object ... 433
18.4.22. POLYGONS and POLYGONS HV Objects ... 435
18.4.23. POLYLINE and POLYLINE HV Objects ... 441
18.4.24. POLYLINES and POLYLINES HV Object ... 444
18.4.25. PRESENTER Object ... 450
18.4.26. RECTANGLE Object ... 453
18.4.27. RECTANGLES Object ... 456
18.4.28. REFRESH TEST Object ... 462
18.4.29. REPORT Object ... 463
18.4.30. SELECTOR Object ... 466
18.4.31. SLIDER Object ... 474
18.4.32. SWITCH Object ... 478
18.4.33. SWITCH SET object ... 486
18.4.34. SYNCHRONIZER Object ... 491

18.4.35. TEXT Object 492
18.4.36. TEXTS Object 496
18.4.37. WORK POINT Object 504
18.4.38. CHART Object 507
18.4.38.1 Chart Service Functions 510
18.4.38.2 Context Menu 511
18.4.38.3 CHART Toolbar 512
18.4.38.4 Keys which Activate the Chart Service Functions 515
18.4.38.5 Mouse-activated Chart Service Functions 520
18.4.38.6 Setting Parameters 520
18.4.38.7 Setting Parameters from Variable Definitions Database 527

19. OPTIONS DEFINED IN ASIX.INI FILE 531

20. ARCHITECT - INTERACTIVE ENVIRONMENT FOR APPLICATION CONFIGURATION 533

21. ASALERT - SYSTEM FOR REMOTE NOTIFICATION OF ALARM EVENTS 535

22. ASAUDIT - SOLUTION FOR SYSTEMS SUBJECT TO STRICT VALIDATION PROCEDURES 537

23. MULTILINGUAL APPLICATIONS 539

23.1. Declaration of Application Languages 539
23.2. Selection of Application Language 539
23.3. Translation Mechanisms 540
23.3.1. Multilanguage Texts 540
23.3.2. Text Table 540
23.3.3. Translation Table 541
23.3.4. Variables Database Substitute Attributes 542
23.4. Change in Coding Pages of Fonts 543
23.5. Handling of Application Components 543
23.5.1. Application Parameters Declared With Use of Multilanguage Texts 543
23.5.2. Operator Actions 544
23.5.3. Scripts 544
23.5.4. Reports 544
23.5.5. Built-in Trends 544
23.5.6. Visualisation Masks 544
23.5.7. Visualisation Objects 545
23.5.8. Tables 545
23.5.9. Alarm System 545

24. ASBASE 547

25. ASCOMM LINK MANAGER 549

26. ASIX4INTERNET - VISUALIZATION AND CONTROL VIA INTERNET 551

27. ASIXCONNECT 5 PACKAGE 553

28. ASTREND PROGRAM 555

29. ASLVIEW PROGRAM 557

30. ASPADTOOLS PROGRAM 559

31. MANAGER OF LOGICAL CHANNELS FOR THE OPC DRIVER 561

32. DISPATCHER MEMO 565

33. AS_POWER_CLOSE 567

34. ZMIANADP PROGRAM 569

35. PATTERN TRENDS 571

36. SCRIPT MODULE 573

37. SIMULATOR 575

38. LIST OF FIGURES 579

39. LIST OF TABLES 583

1. What`s New in asix5

asix - a product of ASKOM company - is a PC software for design and realization of visualization and monitoring for industrial processes, technological lines, machines and devices, monitored and controlled by programmable controllers and other measurement and control systems.

asix belongs to the SCADA (Supervisory Control and Data Acquisition System) software class and executes comprehensive functions of process data collection, processing and visualization, supervisory and direct control, alarm and event recording, data archiving, trend drawing, report generating, data analyses and making process data available within computer networks.

This software system has been created in close collaboration with practicing control engineers and is continuously developed and modified to match users' needs; it is not only a convenient and effective tool for control engineers, but also process data source for MES (Manufacturing and Execution Systems) based on standard techniques. Available since 1995, it is a solution proven already in about 1900 system installations.

asix overrides functional and technical limits set by SCADA systems so far:

- Improved productivity of applications design thanks to elements of batch and interactive on-line parameter setting.
- Built-in application editor in every run-time licence.
- Extended library of technological symbols with a built-in editor.
- Possibility of on-line application parameter setting from a remote station.
- Central management of application update.
- Remote process view on Terminal - type stations.
- Recording of variables course with time resolution in seconds and automatic data archive compression without quantitative limitations.
- Data archiving in daily, monthly and annual files, and also in a database - depending on user's needs.
- Current, historical and pattern charts with dynamic parameter setting and scaling.
- AsTrend module allowing designing, displaying and printing of trends for any set of variables.
- Alarm event log limited only by hard disk drive capacity.
- Automatic alarm and data back-up on spare fixed or removable disk drives (back-up copies creation).
- Built-in interpreter of definable reports in effective language of expressions and formats.
- Bi-directional access to relational databases.
- Context help.
- Ready to operate in multimonitor configurations.
- Built-in hot-redundancy operation mode which improves reliability of workstations.
- Possibility to create systems based on WAN and modem lines.
- Windows application support using the Automation, OPC or DDE engine.
- Possibility to widen practicality of the **asix** system thanks to JScript and VBScript modules.

asix5 is the new version of software package for visualization and industrial process monitoring. New modules enriched the product in solutions expected by our customers.

ARCHITECT - INTERACTIVE APPLICATION CONFIGURATION ENVIRONMENT

Introduction of Architect module is a true breakthrough in application development. It enables a fully visual designing, configuration and ongoing editing of **asix** system application with use of dialogue boxes with system of tabs. The tabs group all options responsible for specific functional areas of the application in an orderly and clear manner and unfailingly lead the designer through the process of correct completion of the required parameters. Architect module, associated with VarDef module responsible for storage of variable definitions databases enables a fully interactive and visual handling of the definitions base, including: creation of base structure and editing its contents.

VARDEF – VARIABLE DEFINITIONS BASE

VarDef manages the databases of Microsoft SQL Server 2000/2005 and of MDB format (Jet / Microsoft Access). To enable migration of application from earlier **asix** versions, the variable definitions data may be converted from Paradox database or text files to MSSQL or Jet base format.

ASAUDIT – SOLUTION FOR SYSTEMS SUBJECT TO STRICT VALIDATION PROCEDURES

Asix5 package is also appropriate for applications, which need to meet specific validation requirements, in accordance with GAMP4, FDA 21 CFR Part 11 regulations applied in pharmaceutical and food processing industry. The module extends the central user log-in and authorization control system with registration of performed controls, operator's action and application integrity control.

ASTREND – GRAPHICAL PRESENTATION OF ARCHIVAL DATA

A series of changes introduced to AsTrend program is an effect of profound redesigning of the module, so as it becomes even more intuitive for the user. The additional asset is the new graphical interface.

APPLICATION DESIGNING AND EXECUTION

Asix5 package also introduces a series of changes and enhanced options, aiming to simplify the design process and to create new process visualization abilities. Among others, a mechanism allowing to control object hiding, their mutual overlaying and object position animation in the process variable value function is included. Many users will be probably happy to hear about such enhancements as ability to operate **asix5** on hardware platforms without an alphanumerical keyboard or the ability to create the graph of single bit value changes for the process variable.

CHANGES IN ASSCRIPTER MODULE

AsScripter in version 1.7 is enhanced with several functions allowing to load series of archival data, and also for reading variable attributes from the variable definitions database and accessing information on current **asix** system user.

NEW ITEMS IN THE COMMUNICATION DRIVERS LIST

AREVA - the driver of serial link RS-485 that allows to exchange data between asix and digital protection devices MiCOM of AREVA; the list of serviced devices includes MiCOM P127 and MiCOM P34x series;
CZAZ - the driver of serial link RS-485 that allows to exchange data between asix system and digital protection devices CZAZ-U and CZAZ-UM of ZEG-Energetyka.
CtMus04 – driver allowing to exchange data between **asix** system and microprocessor-based control devices MUS-04 manufactured by ELEKTROMETAL S.A. from Cieszyn;
MicroSmart – driver exchanging data with MicroSmart controllers of IDEC;
S7_TCPIP – driver used for data exchange with SIMATIC S7 series controllers through Ethernet connection with use of standard computer network card; the product asset is the fact that it does not require the installation of SIEMENS SIMATIC NET software on **asix** system machine, as well as the adaptation of controller software for data exchange purposes;
CtLogo – CtLogo protocol driver is used to exchange data between **asix** system and Logo OBA5 controller from SIEMENS with use of programmator link of the controller;
CtNCP – CtNCP driver is used to exchange data between **asix** system and MN-series controllers from Invensys (former Satchwell).

2. Help System

asix uses standard system help tools.

The use of the help system is identical, as in every other program that operates in Windows system. The help system includes three windows called: ***Contents***, ***Index*** and ***Search***.

The ***Contents*** window enables review of the table of contents of the **asix** documentation with possible opening and browsing of the individual topics.

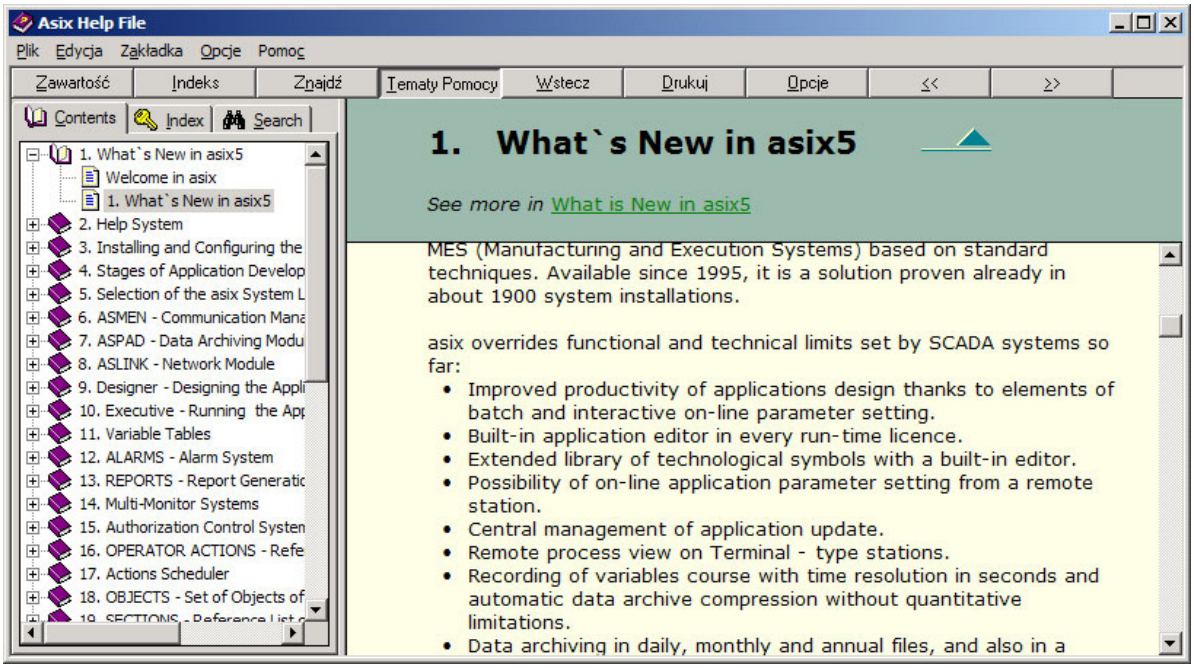


Figure 2-1. Asix Help File - Contents Tab.

The ***Index*** window contains the list of keywords; reference of them can be looked through.

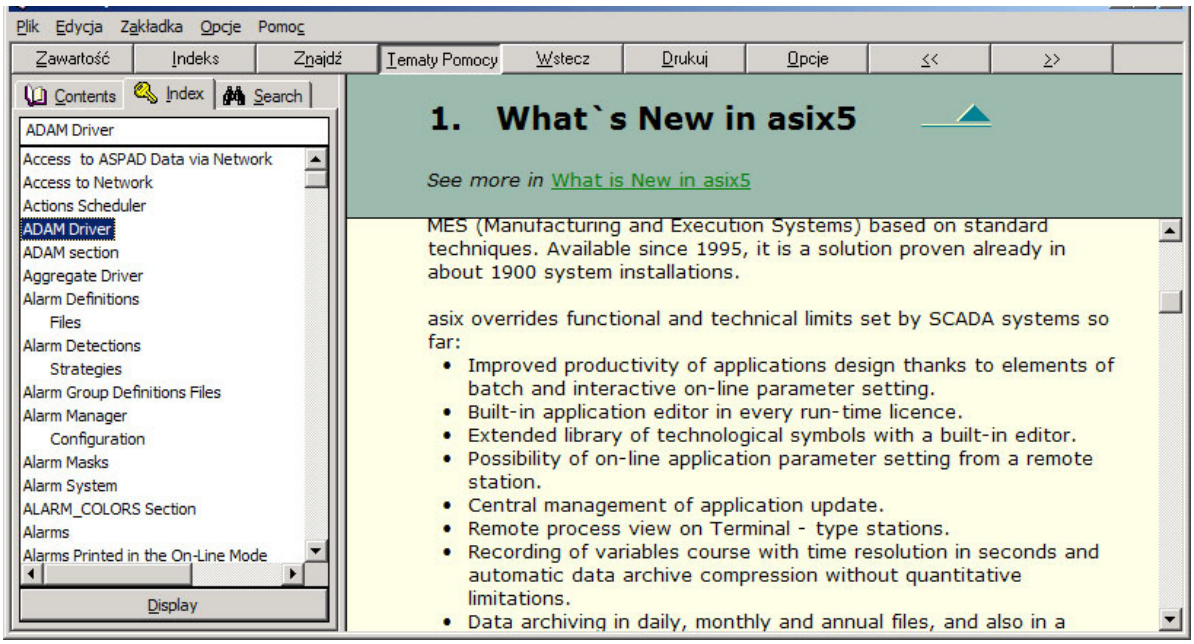


Figure 2-2. Asix Help File - Index Tab.

The ***Search*** window enables searching for any text in the documentation and looking through the selected part of the help, where this text appears.

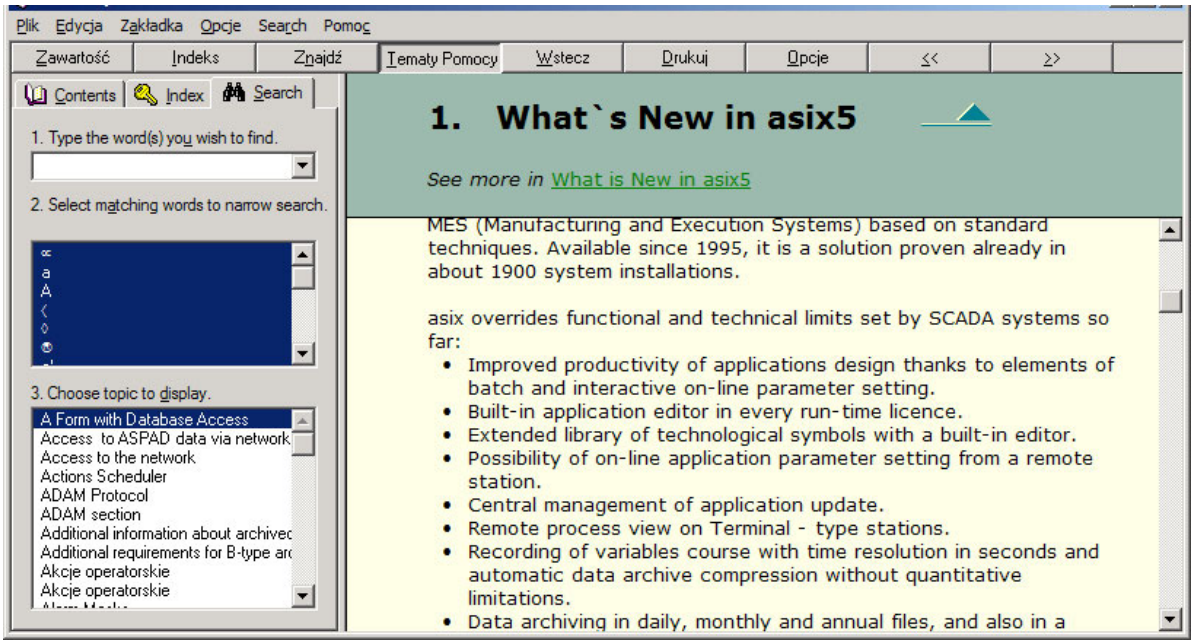


Figure 2-3. Asix Help File - Search Tab.

The **asix** help system can be used separately - with no need to run the **asix** program. For this purpose it is enough to run the asix.hlp file. If the help system is used together with the **asix** program, the help can be obtained at any moment by striking *Alt-F1* keys. The window of context help is then displayed, being referred to the active window (or to the field included in the window) which has been opened before striking

Alt-F1. The help window can also be displayed within the given window by means of clicking the *Help* button (whether exists).

3. Installing and Configuring the asix Software

The **asix** system operates under Windows XP, Windows 2000, Windows NT 4.0 systems.

3.1. Hardware & Software Requirements

The **asix** system may be run on computers with the following minimum hardware equipment:

- operating system: MS Windows 2000/XP, MS Windows Server 2003;
- PC class computer with at least Pentium type processor;
- 256 MB of RAM system memory (or more, depending upon application);
- SVGA video board with 256 colors and 800x600 resolution;
- network board in the case of network installations only;
- mouse or other pointing device with software compatible to the mouse driver interface.

3.2. Installing asix

The installing package includes the following components:

- First Steps user's guide;
- protection key;
- CD-ROM with asix software (with the documentation in the format of hlp and pdf files).

The *First Steps* is a tutorial intended to get acquainted quickly with **asix**. This tutorial enables simple application to be quickly developed and started as well as gives you possibility to know some details of **asix** system's performance and of the methodology of application development. This tutorial contains basic information about installation and main function of **asix**. Reading it, you can build step by step your own simple example application. You can find here information how to enhance your knowledge and answer the most detailed questions. It also allows you to come to believe how quickly and efficiently you can develop SCADA applications in the **asix** environment.

The installation procedure includes the following steps:

- **Installing the protection key**
  Before you start the system, insert the hardware protection key in any parallel port of your computer. The key must be inserted **over the full time** of operation of **asix** programs.

  asix is available in different versions, which differ each other in available operation sets. The key that comes with the software determines the type of installation. The keys are provided with a 4-character code number,

which determines the authorization types in possession. The actual meaning of the code you may find in the price list.

- **Installing the asix software**
  Insert the CD ROM disc into the CD ROM drive. The installation procedure is self-extracting (if problems arise, execute command: asix.exe).

Then, follow the installation program guidelines. It is recommended to close any other Windows programs as far as possible during the installation of **asix**.

STEP 1
After you insert the CD ROM disc into the CD ROM drive, a greeting window will appear.

To proceed, choose *asix installation*.

STEP 2
If Internet modules are installed, it is possible to install:
- .NET Framweowrk 1.1.4322;
- IIS (Internet Information Services) of 5.00 or newer version;
- ASP.NET service has to be switched on in IIS.

If the user decides to install Asix4Internet, the above components will be installed simultaneously with Asix4Internet.

STEP 3
If the **asix** package is being found on the computer; it is necessary to uninstall it or to install the current version to the other directory. There are two variants of installation:
- *full instalation* – requires HASP key;
- *trial installation* – 30-day trial version. After 30 days you must install HASP key and full version.

STEP 4
The next window contains welcome introduction. After pressing the button *Next* you follow to the windows with *License Agreement* and *Readme information*.

Figure 3-1. 'Welcome to the Asix Installation Wizard' Window.

Figure 3-2. 'License Agreement' Window.

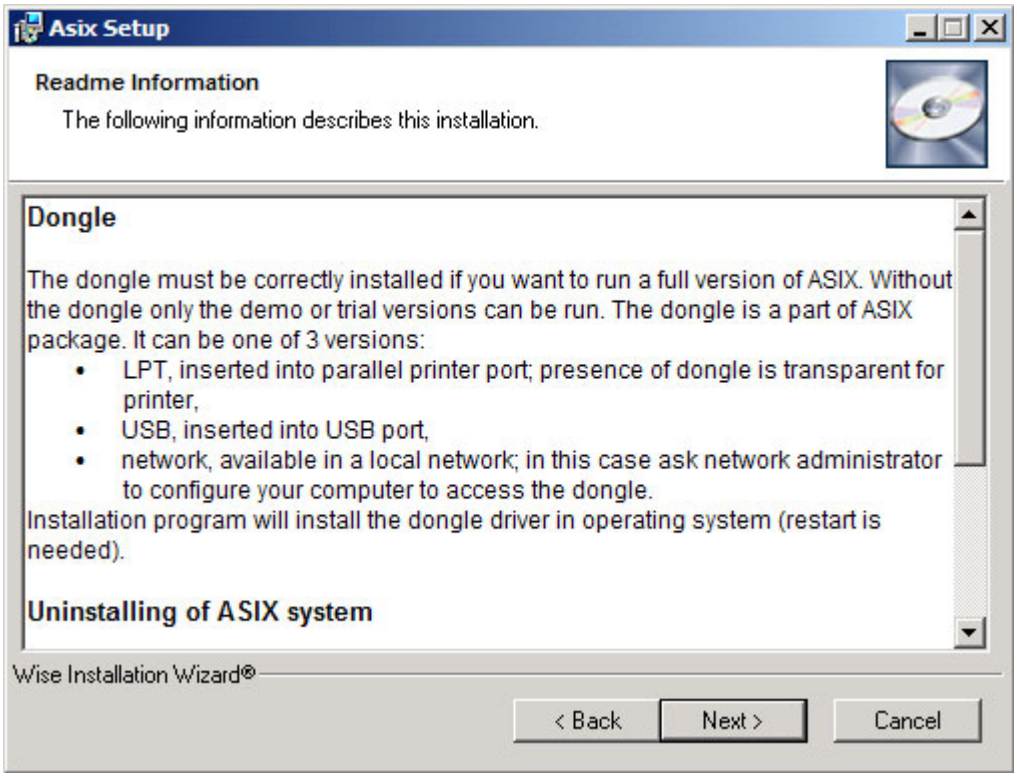


Figure 3-3.'Readme Information' Window.

STEP 5
The following window is to acknowledge the registration.

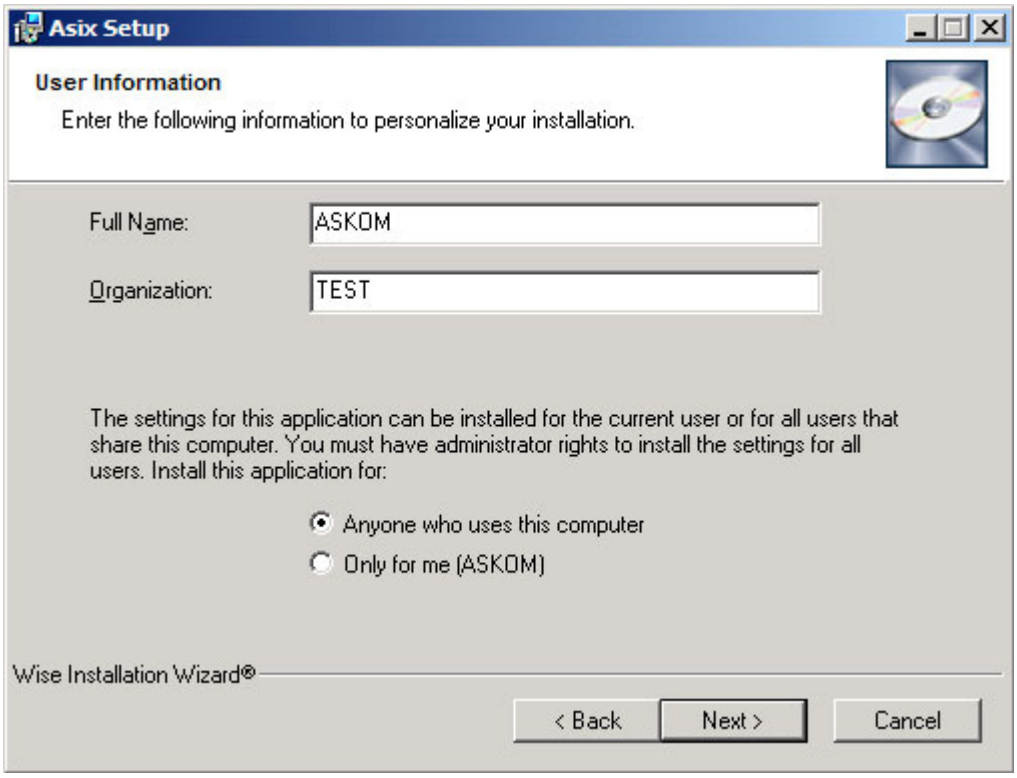


Figure 3-4. 'User Information' Window.

STEP 6
In the next step the installer will run the window, which enables the user to declare the directory.

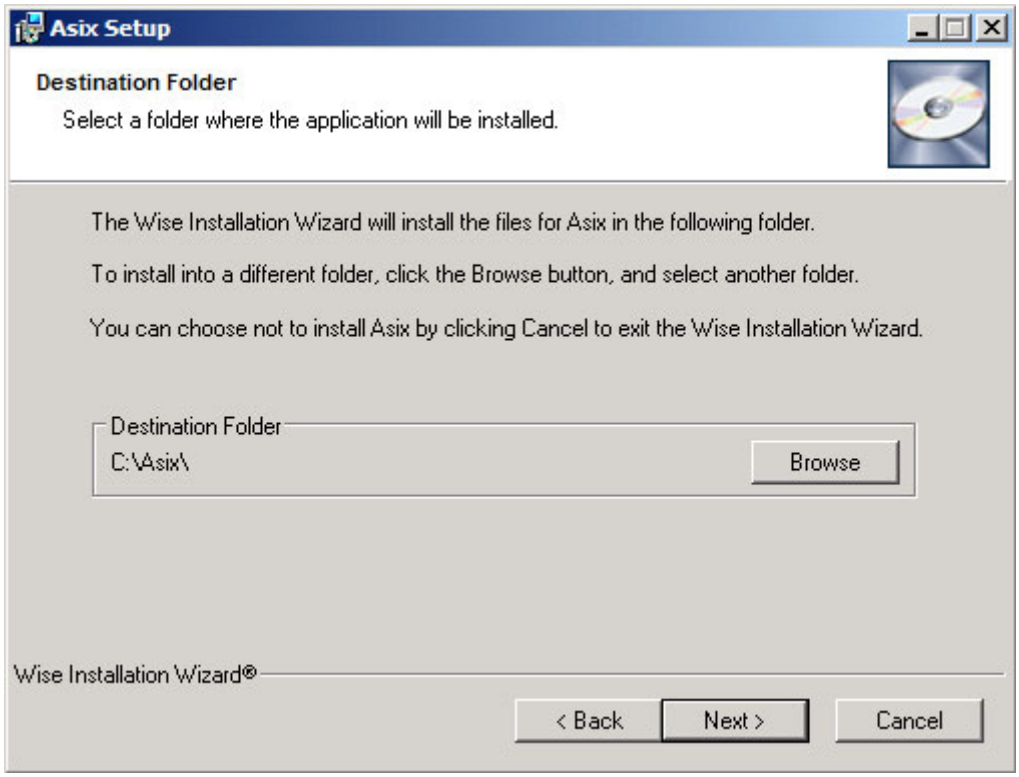


Figure 3-5. 'Destination Folder' Window.

There is a default directory:

<*disk*>:\Program Files \ASKOM\ASIX

where:
<*disk*> is the name of the disk which the Windows system is installed on.

You may change the selected directory using the *Browse* option.

STEP 7
The next window contains list of the system components to be installed. Some elements of the system are selected for the installation by default. The user may freely configure the list of elements to be installed.

- If you intend to use your computer for designing the application, you'd better install all the **asix** components. The correct operation requires the *Programs* to be installed only. If you would like to have the prompt feature, select additionally the *Help* option. *Fonts* contains files including definition of the fonts used for the **asix** version running under DOS for the case of importing the parts of the application developed previously by means of **asix** in its old DOS version.

  The *Programs* option includes:

Visualization package – there are programs and libraries of the **asix** package;

AsAudit - the module extends the central user log-in and authorization control system with registration of performed controls, operator's action and application integrity control.

Drivers – a selection field is used for installing standard and special communication drivers; standard ones are the part of standard **asix** installation but special ones are the separate items on the price list;

AsixConnect - is a package of servers, which are demanded when you intend to use other programs to extend **asix** application functionality (e.g. MS Excel, Access).

- The *Asix4Internet* option should be installed if you intend to use tools for Internet application support.

- If you want to have an access to help system, select *Help*.

- A lot of space is required to install the technical manuals and their viewers (*Manuals*). In the case of insufficient space on the disk you can print the manuals and eliminate them afterwards from the memory.

- If you select *Example applications*, the expandable demo applications will be installed. It is worth to be installed indeed, to see the parts of the example installations and to follow some standard solutions.

- The BDE (*Borland Database Engine*) system is to be installed when it is not installed yet. This component is necessary when process variable data are stored in a database.

STEP 8
The next window notifies about readiness for installation of selected components.

STEP 9
Finally, the window informing about the end of the installation process will be displayed.

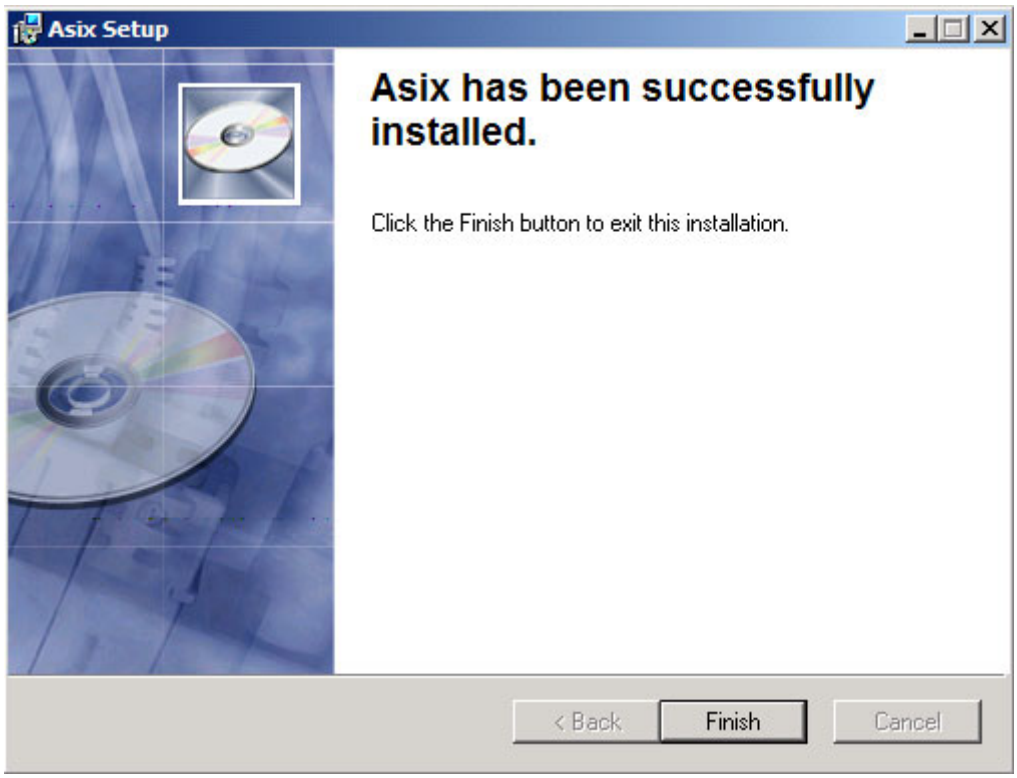


Figure 3-6. Window Informing about asix Installation Performance.

After the **asix** installation, in *Start / Programs* menu the **asix** group, containing particular modules of the **asix** package, will be created.

STEP 10
After the **asix** package has been already installed, the **asix** installation wizard restart displays the *'Application Maintenance'* window with the follwing options:
- Modify – it lets you configure the individual features;
- Repair – preferences stored in the registry may be reset to default values;
- Remove – uninstalls **asix** from the computer.

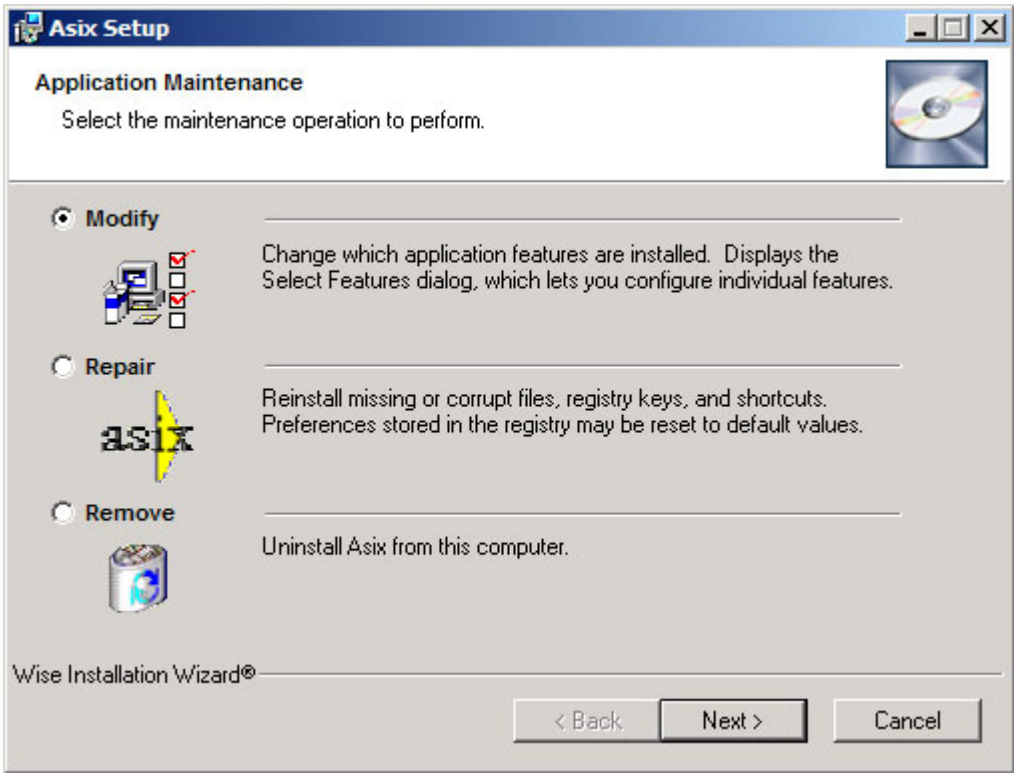


Figure 3-7. 'Application Maintenance' Window.

3.3. Starting asix

asix is started by clicking the previously created application shortcut icon.

To create the shortcut to designer window you should select AS32.exe with the full path from the installation directory in *Properties/shortcut/target element*. In */Start in*: you enter the application's working directory.

NOTICE To run the application with the shortcut, the name of application's *.ini initialization file should be specified as the AS32.exe call parameter.

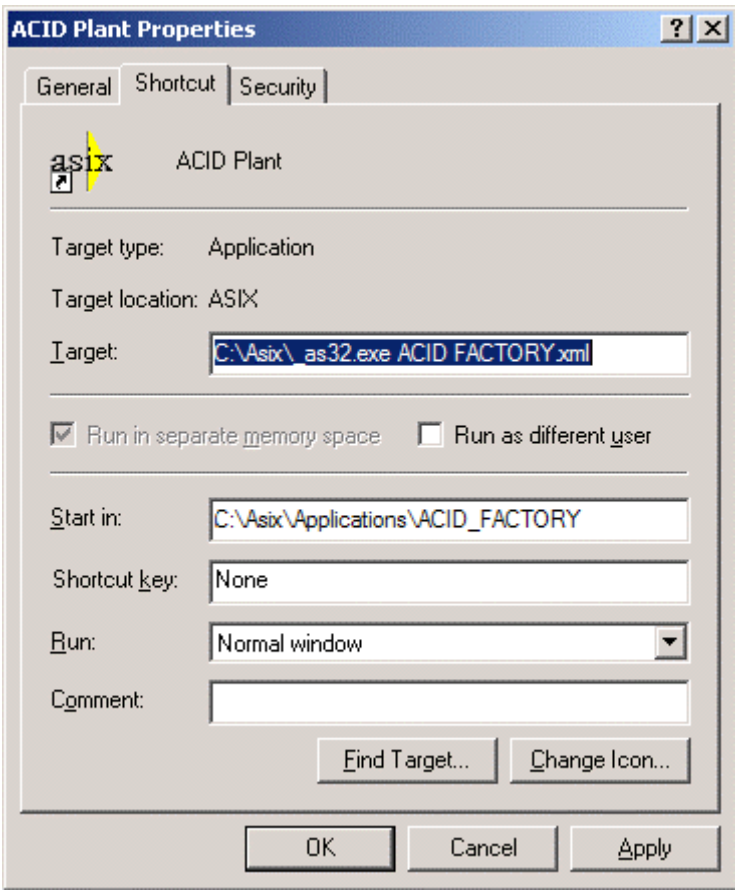


Figure 3-8. Shortcut Properties Window.

An example of shortcut creating:

install directory:	- C:\Asix
initialization file:	- ACID_FACTORY.xml
target:	- C:\AIS\as32.exe ACID_FACTORY.xml
start in:	- C:\ACID_FACTORY

3.4. Uninstallation

Two options give you the possibility to uninstall **asix** system: calling the deinstallation program from the *Control Panel* window (*Add/Remove Programs*) or from the appropriate dialog box – when you run the **asix** package installation file *(see: Figure 3-7. `Application Maintenance' Window),* you should select the *Remove* option.

4. Stages of Application Development

This section describes the successive steps of development of an application in **asix**. In real designing, surely such presented sequence of operations cannot be strictly followed. When expanding application, you will be forced many times to return to the previous points to complete the setting of parameters. Nevertheless, it is worth to be familiar with the guidelines presented below to know the whole method of application development.

Within **asix** you can distinguish certain autonomous modules, which support operation of the main part of **asix** system. The modules are developed as DLL libraries, which are automatically activated when required:

ASMEN - Communication Manager;
ASMEN is responsible for data transmission between the computer on which it has been started up and the PLCs and other process controllers;

ASPAD - Data Archiving Module;
the ASPAD's task is to maintain the long-term archives of all the data retrieving from the process under control;

ASLINK - Local Network Manager;
ASLINK is used for the **asix**'s network installations. It performs information exchange between individual components of the system. It requires the presence of the NEBIOS utility package (referred further as *NETBIOS emulator*). NETBIOS is a component of Windows systems.

Characterized methodology of the **asix** application development includes the following points which will be described in next chapters.

1. Defining Process Data for Asmen Program.
2. Defining Process Data to Be Archived.
3. Creating an XML – application configuration file.
4. Creating a Variable Definitions Database.
5. Creating Visualization Masks.
6. Defining the Way of Application Handling.
7. Defining the Application Operating Parameters.
8. Launching the Application.
9. Network Installations.

For application development you have to use Architect, which is a user-friendly interactive environment offering a brand new manner of application configuration, completely different than in the older versions of **asix** suite. Instead of tiring manual configuration, consisting in creation of applicable entries (sections and items) to the text files, Architect allows a fully visual configuration and ongoing application parameters editing.

4.1. Defining the Process Data for ASMEN Program

The first operation needed to start designing the application is to determine set of the process data as a base on which the application will be developed. All the data necessary to display the status of the industrial process are read by means of the ASMEN program. In order to do it you should set parameters of ASMEN. Two fundamental elements of parameter settings are:

- definition of the read/write transmission channels for the process data, what depends on the choice of hardware equipment to be used for communication with the data source;
- definition of the process data, their addresses, refreshing frequency and method of their processing.

The ASMEN configuration is performed by defining the communication channels in the initialization file and creating the database of variable definitions (**VarDef**) with definitions of process variables, that we will refer to during the process of application development.

Currently it is recommended to prepare the information about process variables with use of tools like spreadsheet or database program and then generate the database of variables of **asix** system from this data. It is also possible to generate the database of variables from the data contained in a text file.

For the VarDef generation you should use the Architect module. Excel spreadsheet allows preparing the full set of information about process variables like: *Name*, *Description*, *Symbol*, *unit*, *Measurering Range*, *Assembly information* etc. This detailed configuration can be yet delayed.

In **asix 5** visualization system, the variable definitions database data can be stored:
- as **Jet** database,
- as **MSQL 2000/2005** database,
- it is possible to use the Excel worksheet directly as the variable base, without the stage of variable generation stage. After the worksheet is loaded, it is no longer used.

Operation, that absolutely has to be performed, is defining the logical names of all used process variables. These names are needed for creation of visual masks.

4.2. Defining the Process Data to Be Archived

The next step is to declare the process variables to be archived. These variables are necessary for calculating the reports as well as for presentation of variable values on masks and trend charts. The ASPAD unit handles all the archived data. In order to set ASPAD's parameters it is necessary to make definition of the set of the ASMEN process data to be archived and the period of their archiving. Parameterization of variables for archiving is included in the database of variable definitions (VarDef). In addition, general program operating parameters should be defined in the configuration file.

4.3. Creating an XML - application configuration file

Creating the XML configuration file for the application is the first step preceding the stage of variable definitions base configuration and application configuration.

The parameter configuring the application operation and declaring the database, which shall be the basis for application operation are saved to **one shared XML file** (*Extensible Markup Language*), placed in the root directory of the designed application.

See more in: Architect user's manual, chapter 3.2. *Creating the new application configuration file*.

4.4. Creating a Variable Definitions Database

In **asix** system the variable definitions database is the space for storing of all information on process variables. Each variable identified by unique *Name* is assigned to one record in the variable definitions database; the fields of this record contain the values of so-called variable attributes. The attribute list includes mandatory, optional and user-defined items. The mandatory attributes are e.g. variable description unit, channel number for communication with the object, archiving parameters. The optional attributes are e.g. limits, format, ranges. Furthermore, the user may also create their own text attributes, individual for each project, e.g. KKS or assembly data.

VarDef system supports Microsoft SQL Server 2000/2005 databases and MDB (Jet/Microsoft Access) databases. In order to migrate the application from earlier **asix** versions, the variable definitions may be converted from Paradox database or text files to the MSQL or Jet database.

Architect module provides a full support of VarDef database in Jet or MS SQL format. In this regard, it combines the functionality of two separate programs applied previously for creating and editing the variable definitions databases: Variable Database Manager and Variable Database Editor (used in the older **asix** versions). Those applications are still included in **asix** suite to enable the use of the older database version.

See more in: Architect user's manual, chapter *2. Handling of the asix application variable definitions databases*.

4.5. Creating the Visual Masks

At the moment when at last the process variable naming and the archived variable naming have been established, you may begin designing the visualization part of the application. In this purpose AS32 program should be started in designer mode.

Under Windows system a shortcut for AS32.exe should be created. (*See*: *3.3. Starting asix*).

The application may be also run in designer mode during Architect module operation with use of *Run Asix System - Constructor* command from *File* menu.

Every application has its fundamental part, the set of visual masks defined for graphic presentation of the controlled process. In general, designing the mask consists of two stages:

- defining the general parameters such as: size, position, activation mode;
- inserting the static and dynamic objects to the mask designed to illustrate the process status and possibly to initiate control operations.

The masks designing is the main activity to be done when creating applications. For details, please refer to the next chapters.

4.6. Selecting the Application Handling Mode

Apart from the creation of visual masks, a very important problem in the application is to ensure its proper logic structure. This procedure is limited to determine the mode of switching between the masks. Although the AS32 program gives certain switching possibilities, they are regarded as auxiliaries only.

The designer may influence the mode of application handling by rendering accessible for operator the so-called "operator actions". Among the variety of actions, the most important ones for this stage of designing there are the opening and closing mask's actions. The designer must define two elements:
full contents of the action,
mode of execution of the action.

In application mode, the actions can be performed by clicking the mouse at the appropriate position on the mask or by pressing the appropriate combination of keys on the keyboard. The first mode requires from the designer to place BUTTON class objects with their actions defined properly on the mask. The another mode requires setting the parameters, which may be done in two ways. If the action is already defined in the BUTTON object, an alternative key combination may be assigned to it at the same time (in the definition of the object). The another method that uses the keyboard requires the so-called keyboard shortcuts sets to be used. These are the application components, that connect various key combinations with operator actions. Such arranged sets can be combined with the masks. The feature of the use of the keys (in both versions) is that the combinations selected are active during the full period of

the mask presence (to which they have been inserted) on the screen. It is unimportant which mask is active.

The next component to be defined is the initial condition i.e. the masks, which are to be opened at the start up of the application:

You should use the *Names of masks which should be opened automatically at application start-up* option in:

Architect program > *Fields and Computers* > *Masks* tab

4.7. Defining the Application Operating Parameters

The last step required to complete the full description of the application is to define the parameters necessary for operating the various AS32 program subsystems. As a rule, such parameters are defined in text files.

There are the components that require setting parameters:

- **Alarm handling system**
  You should define alarm definitions base:

  Architect program > *Databases* > *Alarmbase*

  You should declare paramaters for alarm handling system:

  Architect program > *Fields and Computers* > *Alarms system*

  For more detaled information see:
  Asix – User's Manual, chapter *12. ALARMS – Alarm System*
  Architect user's manual, chapter 3.9. *Configuration of alarm system*

- **Reports**
  Definitions of reports are prepared in text files. Configuration of report definition handling is performed by means of Architect program.

  For more detaled information see:
  Asix – User's Manual, chapter *13. REPORTS – Report Generation Module*
  Architect user's manual, chapter *3.13. Declaration of report definition handling*

- **Trend handling**
  Configuration consists in generating the set of trends with the AsTrend program.

  For more detaled information see:
  AsTrend – User's Manual

- **Application operation parameters**

For more detaled information see:
Architect user's manual, chapter *3. Configuration of asix system application*

4.8. Running the Application

The last step, that has to be made is starting the AS32 program in application mode. Under Windows system a shortcut has to be created with the adequate **asix** icon.

The AS32.exe program start parameter is a name of the **existing** application configuration file.

Three methods of application starting up, which vary in the mode of process data capturing, may be established. They allow testing the application components gradually.

Internally Simulated Data

First stage of testing the created application. To do that, no other program is needed, except for the AS32 program. To disable the operation of ASMEN and ASPAD programs, the *Simulation* parameter should be switched on:

Architect > *Fields and Computers* > *Current data* module > *Standard* tab > *Simulation* parameter

Architect > *Fields and Computers* > *Historical data* module > *Standard* tab > *Simulation* parameter

This method of starting permits the logic structure of the application (mode of mask switching) to be verified.

Data Simulated by ASMEN

The next stage uses the data obtained from ASMEN. Now it is necessary to have fully defined process data. Setting the transmission channels to NONE type causes that ASMEN does not accept on-line data but produces the data simulated internally. Additionally you may set a particular simulation mode for each variable to follow its predicted characteristic. At this stage you may run ASPAD to archive the simulated data.

The data simulation stage performed by ASMEN is the last one to be accomplished directly on the designer's computer without accessing the industrial process. At this stage, testing of compliance between the ASMEN process variable definitions and the process mask variables is mainly carried out.

On-line Data of the Industrial Process

This is the final step of application creation. The ASMEN's transmission channels are configured for on-line retrieving the process data. Obviously, it is related to a proper hardware configuration. What it remains is to verify if the ASMEN process data have been properly defined or not (if they address proper data at the controller memory).

There can be the following optional parameters entered in the application's batch line:

/RESET – it causes operating system to be rebooted in case of the program failure;

/number_of_seconds – it declares, that the program will be started with the delay specified in the parameter. It may be useful, when the program is run from "system autostart" and the time for the operating system to load completely is needed. During the program startup following window will be displayed on the screen:

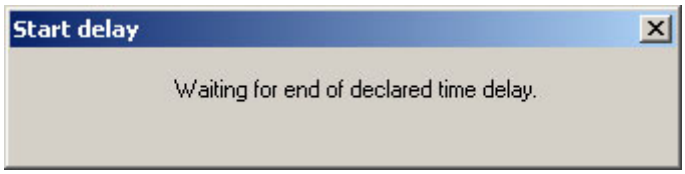


Figure 4-1. 'Start Delay' Window.

/PROTECT – it causes default activation of dynamic protection system at the *as32.exe* program start. See: *9.7. System Protections*.

NOTICE *The parameters can be entered in the application's batch line in optional order.*

4.9. Network Installations

Two main types of stations exist in the **asix** system.

1. **Operator workstation (local station or server) designed for direct handling of the process.**
   Their basic features are as follows:
   - 24 hour operation,
   - direct access to the PLCs controlling the industry process,
   - active archiving the process variables,
   - possibility of process control,
   - on-line alarm handling with acknowledgement facility available.

2. **Remote stations (network terminal) dedicated for supervisory staff.**
   Their basic features are as follows:

- usually periodical functioning only,
- data access by means of network connections with operator stations,
- archived data taken mainly from the operator computer's active archive,
- possibility of process control usually disabled,
- alarms handling by passive browsing of the alarm history only.

Departures from the mentioned above characteristics are possible to allow creation of „hybrid" stations which combine the features of both basic types of stations.

Particular approach is used to the network installations, where a few operator stations are used. Their features are:

- updating, made at the moment of starting of the operator station, is performed to update the process variable archives to the condition of the previously activated operator station;
- archive files of alarms are updated in the similar way;
- alarms are handled in the same way on all operator stations. Acknowledgement, deletion or filtration of the alarm at one station automatically evokes the same action at all the other operator stations.

See description of ASMEN and ASPAD modules and AS32 program handling module in the next chapters for detailed rules of parameter setting for **asix**.

5. Selection of the asix System Language

The **asix** software package possesses the possibility of switching the language of program operation between Polish and English by means of the SelectLanguage.exe program placed in the directory of installed package. The program for language switching is started by the command *Start/Programs/ASIX/**Select Language***.Thanks to such solution the user can quickly and easily determine the language in which the target application will be executed, without the need of manual defining the proper section in the application INI file (as it was in the older version of the **asix** system) or package installation in the required language.

The operation of switching the language of the **asix** system consists in setting options in the following dialogue windows.

First of all, the user is asked for selecting the language version for the SelectLanguage.exe program.

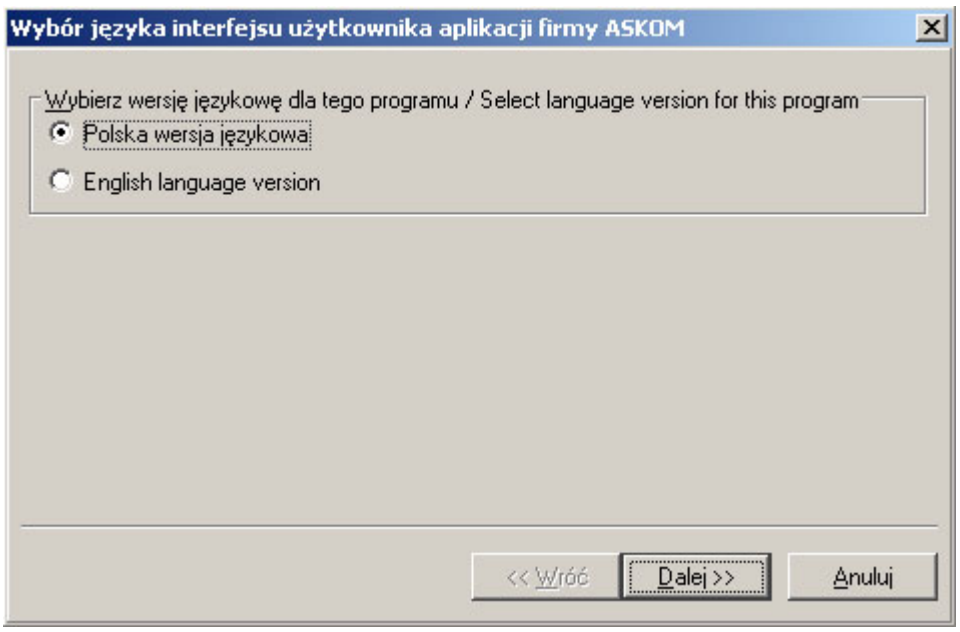


Figure 5-1. 'User Interface Language of ASKOM Applications' Window -1.

The following windows have an informative character.

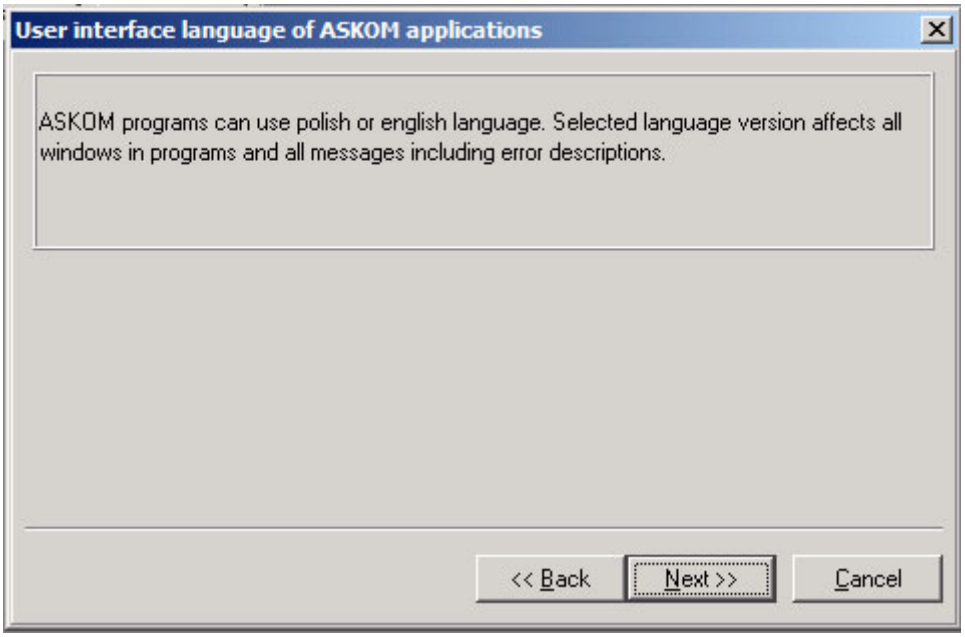


Figure 5-2. 'User Interface Language of ASKOM Applications' Window -2.

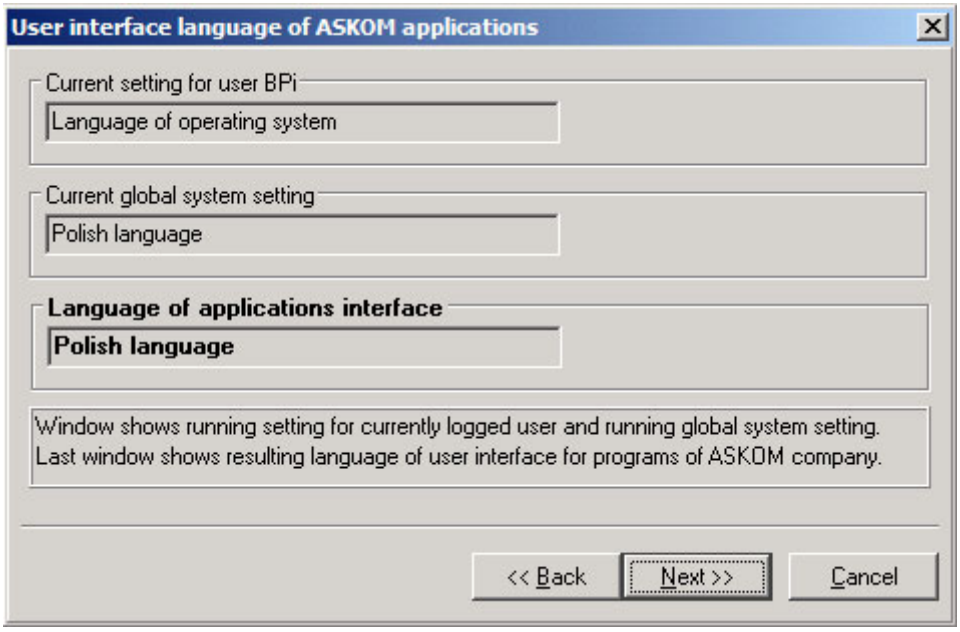


Figure 5-3. 'User Interface Language of ASKOM Applications' Window -3.

The selection of the user interface language of the **asix** system causes the projection of windows (all windows in programs and all messages including error descriptions) in the chosen language. Only the package documentation language is not switched – the documentation in PDF and HLP format files is accessible in the language in which the system was installed.

English application developing demands the **asix** system to be switched to English. Therefore, to make it possible for the user working on the package installed in Polish to develop an English application - the PDF file containing

names of all operator actions, objects, units and functions in Polish along with English equivalents is accessible in the Polish documentation.

The essential meaning has language settings in two below windows - the first window determines the language version for the user who is currently logged in to the operating system, the second window determines operating system settings which will be obligatory for all the other users (if they don't have their own individual settings).

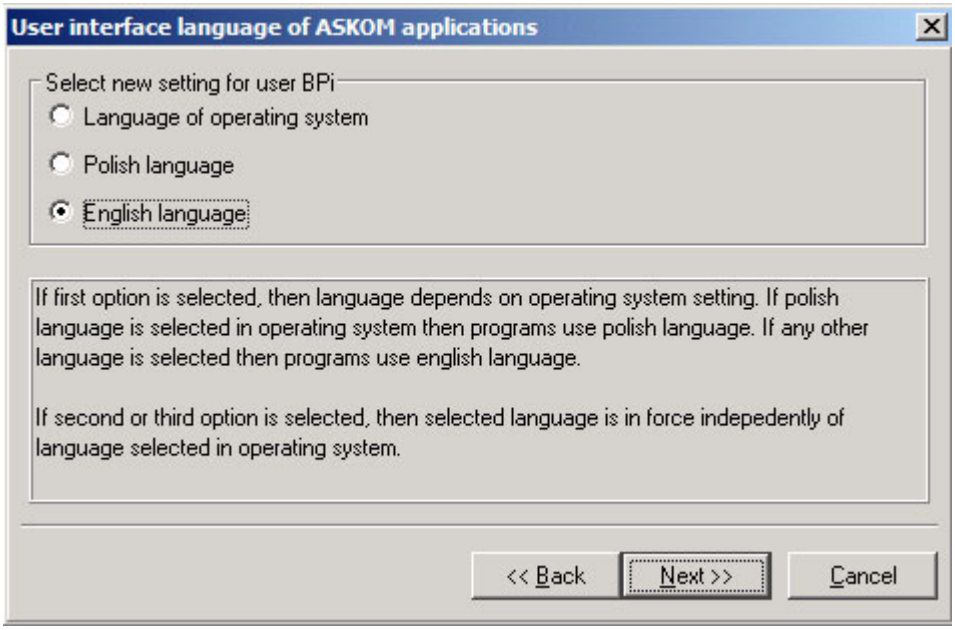


Figure 5-4. 'User Interface Language of ASKOM Applications' Window - 4.

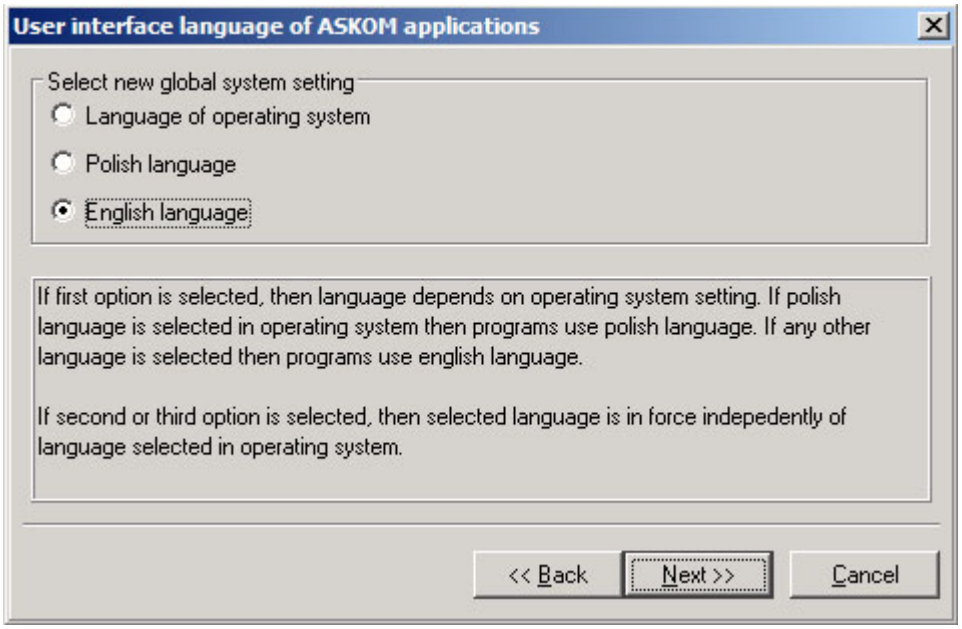


Figure 5-5. 'User Interface Language of ASKOM Applications' Window - 5.

Both cases make it possible for the user to determine the language with use of the following options:
- *Language of operating system*;
- *Polish language*;
- *English language*.

If the first option is selected (*Language of operating system*), then language depends on operating system settings. If *Polish language* is selected in the operating system, **asix** uses the Polish language. If any other language is selected, the **asix** system uses the English language.

If second or third option is selected, then selected language is in force independently of the language selected in the operating system.

At the end, the SelectLanguage program shows new settings for the currently logged user and global system settings that are to be confirmed. The settings for the currently logged user are taken into consideration as first. The system settings are obligatory in case of starting the **asix** program by other users who do not possess individual settings.

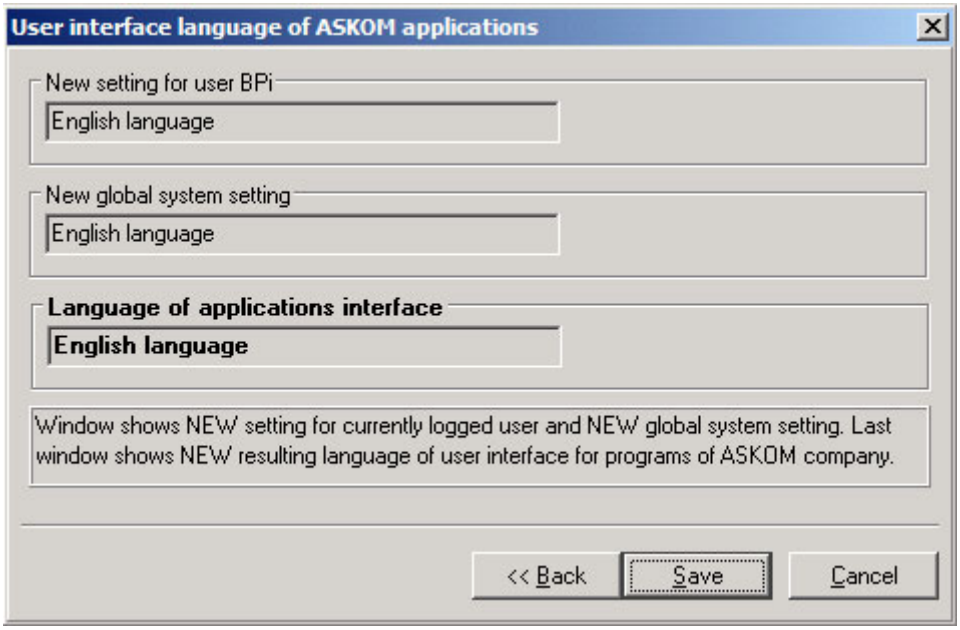


Figure 5-6. 'User Interface Language of ASKOM Applications' Window - 6.

6. Communication Manager ASMEN

Information exchange between a SCADA system and a technological process controller may be performed by means of either serial links or LAN (Local Area Network). Selection of the link type depends on both hardware configuration and requirements that are made to the visualization system in regard to the amount of exchanged data and frequency of data transfers.

Information exchange between **asix** system computers may be performed by means of a LAN. That type of a link will be referred to as a network channel. It makes values of current and archive variables of **asix** available for the computers that are not physically connected to a controller. Of course, at least one computer operating in an ETHERNET network must be physically connected either to the controller or to the network of controllers (e.g. SINECL2), so it would operate as data server for other computers.

The systems that are characterized by a large amount of exchanged information use, as a rule, the Local Area Network, however, for the other systems, serial link interface would meet user's needs.

The designing process of the **asix** system from the point of view of process data availability consists of preparation in text files the following declarations:

transmission channel - is the physical link between the **asix** system computer and the controller located at the plant, defined by the name of communication driver (operating according to the determined protocol) with the list of its particular parameters. The link between **asix** system computers by means of an ETHERNET network can be considered as the transmission channel as well. This link type is mainly used for data transmission to the computers that operate as so-called network terminals, they do not have links to the controllers and do not create their own archive. Specification of a virtual (simulated) transmission channel, that does not need a link to the plant, is also possible. This option is particularly useful on the stage of application design and testing.

process variables - the process variable is the process information that defines status of a given technological parameter, being created on the base of information received from the plant.

In **asix 5** visualization system, the variable definitions database data can be stored:
- as **Jet** database,
- as **MSQL 2000/2005** database,
- it is possible to use the Excel worksheet directly as the variable base, without the stage of variable generation stage. After the worksheet is loaded, it is no longer used.

The process variables, used exclusively for internal application needs and are not related to the technological plant, can be created too.

Based on the above specifications the repository of process variables, being available for all the components of **asix**, is created (i.e. cashe). Values of process variables are stored by the ASMEN module in the operating memory of the computer and are on-line updated by means of their sequential reading from the controlled plant. Information from the object is usually transmitted in raw form, so it need to be initially processed. This processing is performed by means of *calculation functions* declared by the designer for each process variable separately.

All the operations related to the access to process variables are provided in the **asix** system by ASMEN. The ASMEN program supervises process of sequential updating of values of the process variables in the database and interacts in the all operations referred to transmission of the controls to the plant. The direct data exchange between computer and plant controller is performed by drivers, which operate under control of ASMEN. The set of drivers used for the visualization system depend each time on the method of connection of the system to the given plant controller.

Simulation mode is a special mode of ASMEN. This mode is based on the fact, that the values of process variables are not obtained from a physical transmission channel, but they are generated by means of special calculation functions, developed for simulation of plant operation. The simulation mode is particularly useful on the stage of application design, when physical connection to the plant is usually unavailable.

Operating parameters of the ASMEN program are declared in the initialization file. It is a text file that may be modified by means of any text editor that process ASCII files. The initialization files are in general use in **asix** system. The format of these files will be further described in details in another part of the manual.

Declarations of ASMEN parameters have to be set up with use of Architect module. Definition of the following parameters is possible:

- declaration of transmission channels,
- overload warnings (item OVERLOAD),
- ASMEN operation in simulation mode,
- pre-refreshing the variables on the installation stage of ASMEN,
- redundancy of transmission channels,
- declaration of alarm numbers activated in the moment of breaking and restoring the connection in the channel,
- permission for remote reading data from the channel by other channels,
- permission for remote writing to the channel by other computers,
- lock of local writing into the channel,
- time of data validity,
- default cycle of data refreshing,
- path declaration for the TABLE function,
- declaration of the status format compatible to OPC,
- number of access attempts to network variables in the server,
- asynchronous control declaration,
- setting up the priority of the ASMEN threads.

See more in: Architect user's manual, chapter *3.6. Configuration of current data parameters*

6.1. Declaration of Transmission Channels

Declarations of transmission channels enable passing to the ASMEN program the list of drivers used by an **asix** application as well as the operating parameters of those drivers. Declarations of transmission channels need the following parameters:

logic_name - the logical name of a transmission channel; the maximum length of the logical name can't exceed 9 characters; space, comma and tabs are not allowed in the name;

driver_name - the name of the driver of physical link used in the transmission channel; the name NONE, standing for the protocol that simulates a physical link, or the name NETWORK that denotes a network channel can be used;

parameters - the list of parameters, specific for each of protocols of a physical link; the protocol that simulates a physical link (NONE) and the network protocol (NETWORK) need no parameters.

Communication channels are defined with use of Architect module:

Architect program > *Fields and Computers* > *Current data* module

Each channel has its own definition, which is added to *Current data* module by clicking the right mouse button on the *Current data* module and selecting the *Add channel* command:

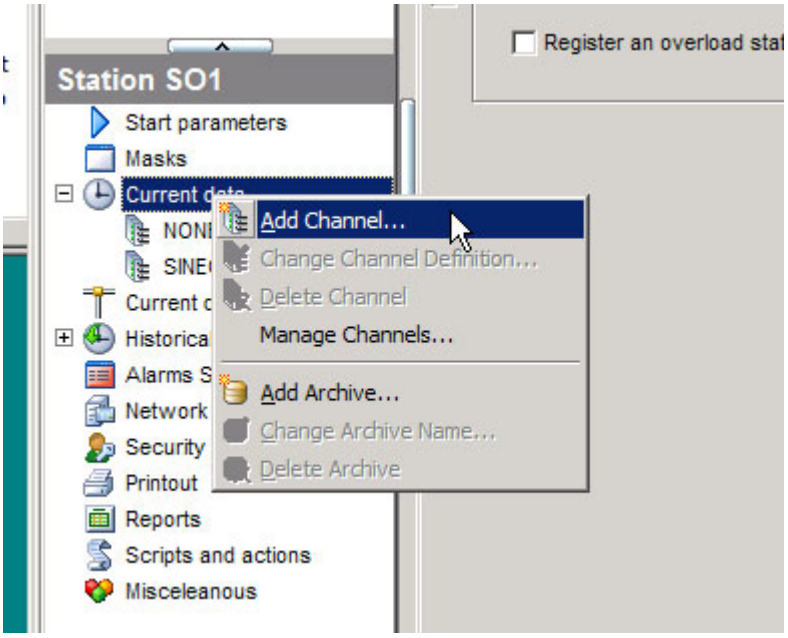


Figure 6-1.Adding new communication channel definition.

See more information: Architect user's manual, chapter *3.6.4. Definition of communication channels*

The **NETWORK** protocol used by the NETSRV.EXE program is a special type of protocol. It does not provide a physical link to a controller, but only connection to any other computer that has such a link. The driver of the network channel, that have established the link, sends a query to all servers, defining the logical name of a channel. The query is responded by only those servers, where such a channel has been previously declared. If more that one server responds the query, the one with the less loaded is chosen to establish the link. If the chosen server is turned off during operation, network link will be established with another server.

The network channel can be used for creating the network terminals that use current variables of **asix**.

The protocol named **NONE**, which is internally served by ASMEN, is a special kind of protocol, which does not provide physical link to a controller. It can be used for the following purposes:

- application testing in simulation mode,
- enabling exchange of information between programs of **asix**; information is exchanged by means of process variables.

Simulation of changing values of process variables that use the NONE protocol can be carried out based on special calculation functions or, on the other hand, by a separated simulation program that modifies values of process variables using the algorithm established by the designer. To use the NONE protocol for application testing, the following are to be done:

- removing declaration of physical transmission channels;
- creating new declaration of transmission channels with logic names same as have been used in original declaration of transmission channels; NONE should be declared as the protocol type;
- after having completed all the tests the original declarations of transmission channels must be restored.

The above procedure may be supported by calculation functions designed for simulation of trends of variable values. These functions provide automatic change of variable values according to a unique algorithm fixed to these function. In the **asix** package there are, at present, two types of simulating calculation functions: *ON/OFF* and *SAW*.

Data exchange by means of process variables of the NONE type channel consists in the procedure, that variable values are obtained from one of **asix** system programs and stored into the repository of process variables used by ASMEN. The other system components (archiving program, visualization objects) can get those data in a normal way (with no respect, that the data are not transmitted from the plant controller). The only component that enables generation of values of variables of NONE-type channel is the visualization object of the *CALCULATOR* class. *CALCULATOR* sets the value of chosen variable on the basis of cycle calculation of any arithmetical expression.

No additional parameters are declared for the item that describes the NONE-type protocol.

Array variables from the NONE channel are treated as one variable.

There is the possibility to declare the method of the access to variables that relies on associating the value for variables in the effect of executing the saving operation. The operation of saving requires giving not only the value but also the time and the status of variables. It is necessary, in this case:
- to declare a channel with the NONE driver assigned to this channel,
- to select the *Writing data and status* parameter.

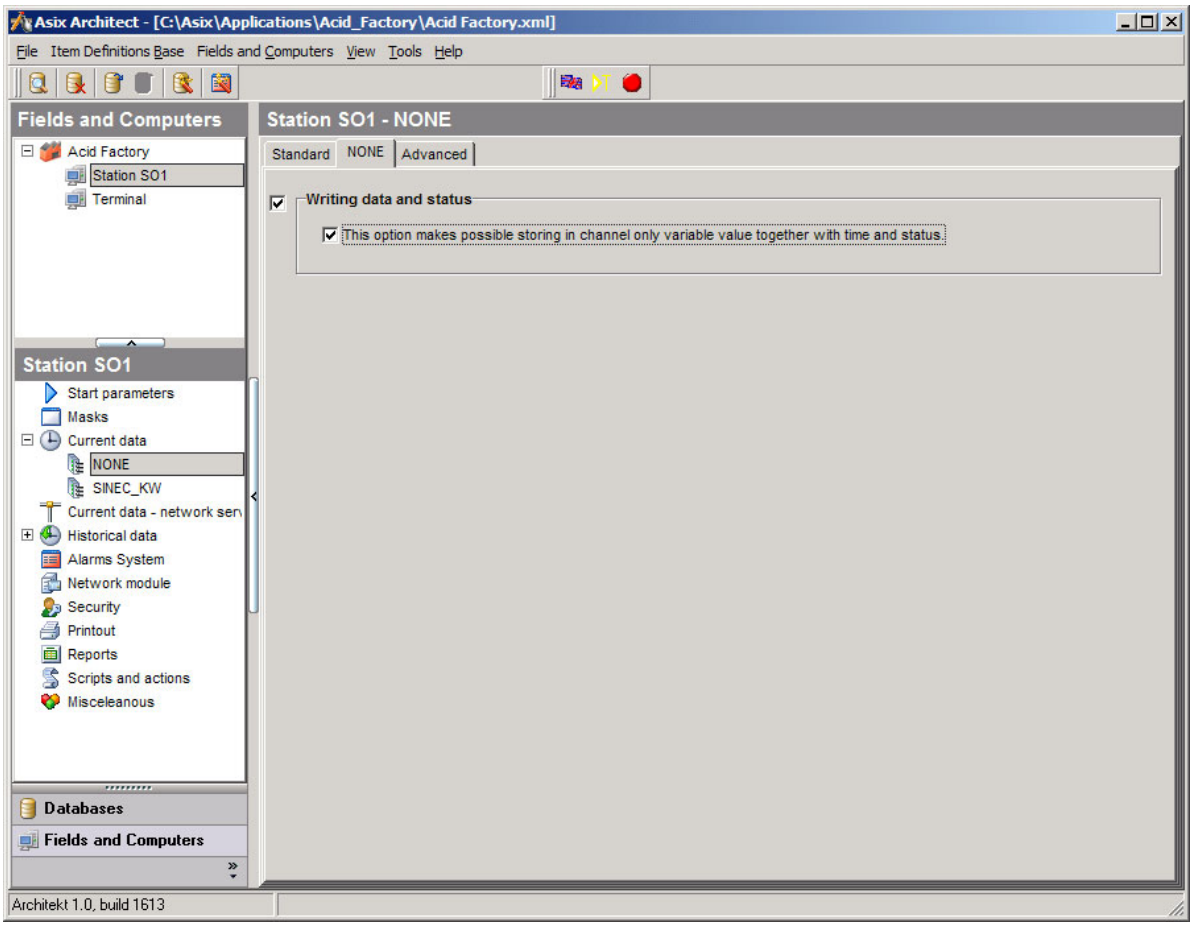


Figure 6-2. 'Writing data and status' parameter declaration.

This solution was created to make import of variables from outside applications (e.g. spreadsheet) and transfer them to the archive of the ASPAD module possible.

NOTICE ***At the moment, when ASMEN starts running it is required, that drivers of all protocols declared in the initialization file would be stored in the directory, where the asix system starts.*** *The driver that simulates the physical link (NONE) is built in the ASMEN program. The loading procedure of individual drivers is defined with the description of drivers available for* ***asix****.*

6.2. Declaration of Process Variables

NOTICE ***Declarations of process variables are received from VarDef (the Variable Definitions Database of the asix system).***

The parameters defining variables in VarDef are located in the fields named as follows:

Parameter	**Field**
Name	Name
Description	Description
Address	Address
Channel	Channel
Elements count	ElementsCount
Sample rate	SampleRate
Conversion function	ConversionFunction
Parameters of the calculation function:	
Conversion function range from	ConversionFunctionRangeFrom
Conversion function range to	ConversionFunctionRangeTo
Measurement range from	MeasurementRangeFrom
Measurement range to	MeasurementRangeTo

The whole list of system attributes of the variable definitions database you can find in Architect user's manual, *Appendix 1*.

When declaring the process variables their type are not defined. The type of process variable in a raw form (read or write to the controller) results from the transmission protocol and symbolic address. The type of process variable in a converted form (transferred by ASMEN to application or from application to ASMEN) results from the conversion function being used. The process variables may take one of the following forms:

- 16-bit unsigned number (WORD),
- 16-bit signed number (INTEGER),
- 32-bit unsigned number (DWORD),
- 32-bit signed number (LONG),
- floating-point number (FLOAT).

The ASMEN module uses the following parameters of variables:

variable_name - unique symbolic name of process variable identifying the variable for all the components of **asix** (name length can't exceed 15 characters and the name must not include spaces, colons, parentheses of any type and commas);

variable_description - any text containing, e.g. technological description of process variable (commas are not acceptable);

address - symbolic address, the form of which is specific for each of drivers of **asix**; it is used by the driver in order to locate the variable in the controller;

channel - logical name of a transmission channel (one of the names given in the declaration of transmission channels);

quantity	- number of items included in a process variable (the variable may be an array), e.g. for data blocks DB it is necessary to specify the number of DW data words, the values of which, in the form of array, will be assigned to the process variable (for protocols of SIMATIC controllers); in a default case the number of items is set to 1;
rate	- rate of updating the value of process variable (in seconds);
function	- name of conversion function that is designed for converting the value received from the controller into the value transferred to components of **asix**; while initiating the process variables the compatibility of the type of variable retrieved from the controller (the variable is defined by the driver) with the type of input variable of the conversion function is checked; in the case of disagreement the process variable is not initiated; it is allowed to use INTEGER, WORD, DWORD and LONG types interchangeably;
parameters	- the list of conversion function parameters that are assigned to the given variable; in the case the list don't include all parameters the default values, specific for each of conversion functions, will be assigned to the parameters which have been omitted.

A symbolic address is used for unique definition of variable in the controller, the value of which will be assigned to the process variable in the **asix** system. On the basis of the symbolic address the type of the variable and its localization in the controller memory are defined. The way the address is created is specific for the channel type in which the variable is located.

6.3. Running the ASMEN Module

ASMEN is loaded automatically while the **asix** start-up and its operating parameters are read the application configuration file. Parameters setting for ASMEN must take into account all the variables used by the system.

6.4. Entries in Configuration File

ASMEN parameters are set up in the configuration file of an **asix** application.

See: Architect user's manual, chapter 3.6. *Configuration of current data parameters*

6.5. Conversion Functions

ANALOG
ANALOG_FP
ANALOG12STATUS
ANALOGWORD_FP
BIT0_Z_N
CAMAC6_1
CAMAC6_2
TIME
TIME_SEC
TIME_MSEC
CIRCLE1
DATETIME
DATETIME_MEC
DW2FP
GRADIENT
QUOTIENT
COUNTER
MASK
MAXIMUM_OF_N
MINIMUM_OF_N
FACTOR_KG
FACTOR_FP
FACTOR_DW
FACTOR
FACTOR_INT
FACTOR_LONG
NEGBIT_BYTE
NEGBIT
NEGBIT_DWNEGBIT_DW
NOTHNG NOTHING
NOTHNG_DW
NOTHING_DD
NOTHING_FP
NOTHING_BYTE
NOTHING_INT
NOTHING_LONG
NOTHING_KG
NOTHNG_TEXT
ON/OFF
SAW
SAW_FP
PERCENT
PERCENT_FP
SHIFT_L
SHIFT_P
AVERAGE
AVERAGE_OF_N
CONSTANT
CONSTANT_FP
STATUS
SUM_OF_N
SLIDER
SLIDER1
SLIDER1_FP
TABLE
WAVERAGE_OF_N
DATA_SCOPE

Calculation functions are used by the ASMEN module when converting values of raw process variables into real form. Each variable is assigned a calculation function declared separately. Some of available calculation functions allow to simulate the variable values according to given algorithm.

The **asix** system includes the following conversion functions:
ANALOG, ANALOG_FP - linear conversion
ANALOG12STATUS - special linear conversion
ANALOGWORD_FP - special linear conversion
BIT0_Z_N - concatenation of word from zero bits of n variables
CAMAC6_1, CAMAC6_2 - special handling of the CAMAC interface
TIME - provides conversion of value of the process variable that have been read from a controller
TIME_SEC - provides conversion of value of the process variable that have been read from a controller
TIME_MSEC - provides conversion of value of the process variable that have been read from a controller
CIRCLE1 - bit-to-number conversion
DATETIME - conversion of number of seconds to a string
DATETIME_MEC - conversion of time of a MEC counter to a string
DW2FP - conversion of variables of DWORD type to variables of FLOAT type
GRADIENT - gradient calculation
QUOTIENT - scaling
COUNTER - conversion from BCD format
MASK - logical product of variable values and function parameter
MAXIMUM_OF_N, MINIMUM_OF_N - maximum value of n variables
FACTOR_KG - conversion of floating-point types
FACTOR_FP - conversion of floating-point types
FACTOR_DW - conversion of floating-point types
FACTOR,FACTOR_INT,FACTOR_LONG - linear conversion
NEGBIT_BYTE,NEGBIT,NEGBIT_DW - inversion of selected bits in byte, in word and in double word respectively
NOTHING, NOTHNG_DW, NOTHING_DD, NOTHING_FP - transmission of value without changing

NOTHING_BYTE, NOTHING_INT, NOTHING_LONG - transmission of value without change
NOTHING_KG, NOTHNG_TEXT - transmission of value without change
ON/OFF - simulation of binary signals
SAW, SAW_FP - simulation of sawtooth trends
PERCENT, PERCENT_FP - conversion to percentage value
SHIFT_L - shift left of variable value
SHIFT_P - shift right of variable value
AVERAGE - average of moving-window type
AVERAGE_OF_N - actual value of arithmetic average of variables list
CONSTANT, CONSTANT_FP - assigning constant value to an ASMEN variable
STATUS - current variable status
SUM_OF_N - sum of values of n variables
SLIDER, SLIDER1, SLIDER1_FP - auxiliary function for the visualization objects of the SLIDER class
TABLE - non-linear conversion
WAVERAGE_OF_N - actual value of weighted average of variable list
DATA_SCOPE - conversion of process variable read from a controller

All conversion functions of **asix** are described below. For each conversion function the following properties are assigned:

- type of read/write variable from/to a controller (type of PLC variable);
- type of a variable passed from/to the user's program (type of PC variable);
- the list of parameters (together with their types and default values) in order of their appearance in a variable declaration line.

ANALOG

The function ANALOG enables conversion of the process variable read from a controller according to the formula:

*PCVal = MinPCVal + (PLCVal - MinPLCVal) * k*

and conversion of the process variable value that is to be sent to a controller according to the formula:

PLCVal = MinPLCVal + (PCVal - MinPCVal) / k

The *k* factor is calculated in the following way:

k = (MaxPCVal - MinPCVal) / (MaxPLCVal - MinPLCVal)

type of PLC variable - INTEGER
type of PC variable - INTEGER

parameters:
MinPLCVal - INTEGER (default value - 0)
MaxPLCVal - INTEGER (default value - 2048)
MinPCVal - INTEGER (default value - 0)
MaxPCVal - INTEGER (default value - 2048)

ANALOG_FP

The function ANALOG_FP provides conversion of the process variable value read from a controller according to the formula:

*PCVal = MinPCVal + (PLCVal - MinPLCVal) * k*

and conversion of the process variable value that is to be sent to a controller according to the formula:

*PLCVal = MinPLCVal + (PCVal - MinPCVal) * k*

The *k* factor is calculated in the following way:

k = (MaxPCVal - MinPCVal)/(MaxPLCVal - MinPLCVal)

type of PLC variable - INTEGER
type of PC variable - FLOAT

parameters:
MinPLCVal - INTEGER (default value - 0)
MaxPLCVal - INTEGER (default value - 2048)
MinPCVal - FLOAT (default value - 0.0)
MaxPCVal - FLOAT (default value - 2048.0)

NOTICE *The ANALOG_FP function operates like NOTHING_FP function when it is used for variables of the NONE channel.*

NOTICE *The function doesn't execute the operation of saving if it is used for the process variable of the NONE channel, for which the operation of saving with time and status WASN'T declared.*

ANALOG12STATUS

The function ANALOG12STATUS enables conversion of the process variable read from a controller according to the formula:

*PCVal = MinPCVal + (PLCVal - MinPLCVal) * k*

And conversion of the process variable value that is to be sent to a controller according to the formula:

*PLCVal = MinPLCVal + (PCVal - MinPCVal) * k*

The *k* factor is calculated in the following way:

k = (MaxPCVal - MinPCVal)/(MaxPLCVal - MinPLCVal)

type of PLC variable - INTEGER
type of PC variable - FLOAT

parameters:

MinPLCVal - INTEGER (default value 0)

MaxPLCVal - INTEGER (default value 2048)
MinPCVal - FLOAT (default value 0.0)
MaxPCVal - FLOAT (default value 2048.0)

NOTICE It is not necessary to take into account the fact that the read value of an analog is shifted by 3 bits to the left.

NOTICE *The function is used for devices operating 12-bit values of measurements, in which the bits of measurement status are stored in three the least significant bits (data in the SIEMENS format with diagnostic within the status word), e.g. for SIMATIC S5 , BECHOFF, WAGO modules.*

NOTICE *The function operates like the ANALOG_FP function, but additionally the status adequate to the state of bits read from the module AI is set.*

a/ *bit1 = 0, bit0 = 1 – overflow of the upper range; the OPC_LIMIT_HIGH status is given back (or AVD_BAD, if ASMEN is not in OPC mode);*
b/ *bit1 = 1, bit0 = 1 – overflow of the lower range; the OPC_LIMIT_LOW status is given back (or AVD_BAD, if ASMEN is not in OPC mode);*
c/ *OPC_QUALITY_GOOD status is given back for the other combinations of status bits (or AVD_GOOD, if ASMEN is not in OPC mode).*

ANALOGWORD_FP

The function ANALOGWORD_FP enables conversion of the process variable value read from a controller according to the formula:

*PCVal = MinPCVal + (PLCVal - MinPLCVal) * k + offset*

and conversion of the process variable value that is to be sent to a controller according to the formula:

*PLCVal = MinPLCVal + (PCVal – MinPCVal - offset) * k*

The k factor is calculated in the following way:

k = (MaxPCVal - MinPCVal)/(MaxPLCVal - MinPLCVal)

type of PLC variable - INTEGER
type of PC variable - FLOAT

parameters:
MinPLCVal - INTEGER (default value 0)
MaxPLCVal - INTEGER (default value 2048)
MinPCVal - FLOAT (default value 0.0)
MaxPCVal - FLOAT (default value 2048.0)
offset - FLOAT (default value 0.0)

NOTICE *The ANALOGWORD_FP function operates like NOTHING_FP function when it is used for variables of the NONE channel.*

NOTICE *The function doesn't execute the operation of saving if it is used for the process variable of the NONE channel, for which the operation of saving with time and status wasn't declared.*

☑ *BIT0_Z_N*

The function BIT0_Z_N creates a 16-bit unsigned number (WORD), where the values of individual bits are equal to the value of least significant bits(bit 0) of actual values of the variables being specified in the list. The bit no. 0 is set from a value of the variable on the first place in the list, bit no. 1 – of the variable on the second place and so on. The number of parameters is limited to 16.

type of PLC variable - BYTE, WORD, DWORD
type of PC variable - WORD

parameters:

k - minimal number of variables have been read correctly from a controller, for which the validity of function evaluation is recognized; the value of 0 means that all the variables must have been read correctly. If the number of variables that have been read correctly is lower than declared, the function returns a random value with an error status.

Val_1 .. Val_*n* names of ASMEN variables separated by a space, n<=16

NOTE *In case of declaring a variable of **FLOAT** type as one of Val_i variables, before executing a function the variable value is converted to WORD (the function makes sense for binary variables returned by driver as FLOAT).*

☑ *CAMAC6_1*

The function CAMAC6_1 enables conversion of the process variable value that have been read from a controller according to the formula:

for PLCVall >= 2048

PLCVal = (PLCVal - 2048)/2048

for PLCVall < 2048

PLCVal = (2048 - PLCVal)/2048

*PCVal = PLCVal * (MaxPCVal – MinPCVal) + MinPCVal*

type of PLC variable - INTEGER
type of PC variable - FLOAT

parameters:

MinPLCVal - INTEGER (default value 0)
MaxPLCVal - INTEGER (default value 4096)
MinPCVal - FLOAT (default value -75.0)
MaxPCVal - FLOAT (default value 300.0)

CAMAC6_2

The function CAMAC6_2 calculates an average from samples of the process variable values read from a controller according to the formula:

for PLCVall >= 2048

$$PLCVal = (PLCVal - 2048)/2048$$

for PLCVall < 2048

$$PLCVal = (2048 - PLCVal)/2048$$

$$PCVal = PLCVal * (MaxPCVal – MinPCVal) + MinPCVal$$

The average is calculated for *Period* time period (given in minutes) and the result is regarded as correct if the percentage rate of correctly read samples is equal to *Rate*.

type of PLC variable - INTEGER
type of PC variable - FLOAT

parameters:
MinPLCVal - INTEGER (default value 0)
MaxPLCVal - INTEGER (default value 4096)
MinPCVal - FLOAT (default value -75.0)

TIME

The function TIME provides conversion of the process variable value that has been read from a controller (considered as a time counter in STEP5 format) to the number that represents time in units defined by a parameter of TIME function.

When the value of the system variable is sent to a controller, conversion of the number representing the time in user's units to the time counter in STEP5 format is performed.

The user time formats are listed below:
type 0 - time in 0.01s units
type 1 - time in 0.1 units
type 2 - time in seconds
type 3 - time in minutes
type 4 - time in hours

type of PLC variable - WORD
type of PC variable - WORD

parameters:
Time format - WORD (default value - 2)

TIME_SEC

The function TIME_SEC provides conversion of the *WORD* process variable value read from a controller (as a number of seconds) to the string of ASCII signs of the following format:

mm:ss

where:
- mm - minutes
- ss - seconds

The function TIME_SEC converts during the saving process the string of ASCII signs of the following format:

mm:ss

where:
- mm - minutes
- ss - seconds

to the number of seconds of WORD format.

NOTICE *The process variable using the function may not belong to the NONE channel.The function may not be used for array variables.*

TIME_MSEC

The function TIME_MSEC provides conversion of the *DWORD* process variable value read from a controller (as a number of milliseconds) to the string of ASCII signs of the following format:

hh:mm:ss

where:
- hh - hours
- mm - minutes
- ss - seconds

The function TIME_MSEC converts during the saving process the string of ASCII signs of the following format:

hh:mm:ss

where:
- hh - hours
- mm - minutes
- ss - seconds

to the number of milliseconds in DWORD format.

NOTICE *The process variable using the function may not belong to the NONE channel.The function may not be used for array variables.*

CIRCLE1

The function CIRCLE1 provides conversion of the *WORD* process variable value read from a controller to one of 16-bit numbers (without the sign) of the array assigned to a given process variable. The contents and size of the array is determined on the basis of the process variable declaration. Conversion consists in searching of the first non-zero bit in the number that has been read from the controller. The first non-zero bit position is then used as an index for

reading from the table. Searching the non-zero bit starts from the bit defined in declaration of a variable and is proceeded towards LSB (less significant bit). The number of tested bits is given in declaration of the variable. In the case, when a given word equals zero, the value of the last parameter in declaration of the process variable is returned as function value.

When sending to a controller, the function CIRCLE1 searches the array assigned to the process variable in order to find the pattern identical to the one passed to the function. When the pattern is found, the 16-bit number that has 1 on position referred to the index of found pattern and zeros on the other positions is sent to the controller. The pattern on the last position in the array is an exception - in this case zero is sent to the controller.

In the case, when the pattern is not found, the function returns error.

type of PLC variable - WORD
type of PC variable - WORD

parameters:
number of first bit, from which the search starts - WORD
number of searched bits – WORD
successive entries of pattern array (max. 16+1 - the last one for 0-entries) – WORD

DATETIME

The function DATETIME converts number of seconds transferred in the number of DWORD type to an ASCII string of the following format:

DD-MM-YYYY:HH:NN:SS

where:
DD - two digits of day
MM - two digits of month
YYYY - four digits of year
HH - two digits of hour
NN - two digits of minutes
SS - two digits of seconds
b - space

It may be used exclusively for the NONE driver.
The function has no parameters.

DATETIME_MEC

The function DATETIME_MEC converts the MEC counter time given as a number of seconds transferred in a string of 17 bytes to an ASCII string of the following form:

YYYY-MM-DD HH:MM

The MEC time is transferred by means of index 0.
The function has no parameters.

☑ ***DW2FP***

The function converts variables of DWORD type to variables of FLOAT type.

☑ ***GRADIENT***

The GRADIENT returns the variable value calculated according to the formula:

$y = \{(t_2 - t_1) / T\}*m$

where:

t_2 – current value calculated as weighted average
t_1 – value from the moment t_0-T calculated as weighted average
T – period, for which the gradient is calculated (in seconds)

weighted average $t = t_{-1}*(1-k) + t_b*(k)$

k – weight factor in range (0 – 1)
t_{-1} – value of t calculated in the previous cycle
t_b – present value
m – multiplier

type of PC variable - FLOAT

parameters:

T	- WORD
k	- FLOAT
m	- FLOAT

A source value for the GRADIENT function can be both the raw variable in the controller and the variable of **asix** declared in ASMEN.

EXAMPLE

An examplary declaration of variable calculated by means of the GRADIENT function.

VARIABLE1,gradient of the variable in a controller,ED100.0,CHANNEL,1,1,GRADIENT,15,0.1,60

Meaning of parameters:

Variable address - DB 100, DW0
Period T - 15 seconds
Factor k - 0.1
Multiplier m - 60 seconds
VARIABLE1 – it is a calculated trend of change in controller units per minute of a variable of address ED100.0.

NOTICE *The variable in a controller must be given in real units.*

EXAMPLE

An examplary declaration of the **asix** system variable, which is calculated with use of the GRADIENT function:

VARIABLE2,gradient of a variable,,NONE,1,1,GRADIENT,15,0.1,60,VARIABLE3

Meaning of parameters:

Period T - 15 seconds
Factor k - 0.1
Multiplier m - 60 seconds
Source variable - VARIABLE3
VARIABLE2 – is a calculated trend of changes of the VARIABLE3 variable in units of VARIABLE3 per minute.

QUOTIENT

The QUOTIENT function enables conversion of the process variable value that has been read from a controller according to the formula:

PCVal = PLCVal / k

and conversion of the process variable value that is to be sent to a controller according to the formula:

*PLCVal = PCVal * k*

where ***k*** (QUOTIENT) is the parameter of this function.

type of PLC variable - INTEGER
type of PC variable - FLOAT

parameters:
Quotient - FLOAT (default value - 10.0)

COUNTER

The COUNTER function enables conversion of the process variable value that has been read from a controller (considered as a 3-decade BCD counter) to the 16-bit unsigned number.

When a system variable value is sent to a controller, conversion of 16-bit unsigned number to 3-decade BCD counter is made.

type of PLC variable - WORD
type of PC variable - WORD

no parameters

AND

The AND function performs an operation of logical AND on the variable value that has been read from a controller and on the parameter of the AND function that is given in the variable declaration.

When writing to a controller the variable value is not modified.

type of PLC variable - WORD
type of PC variable - WORD

parameter:
number in hexadecimal format (default value FFFF).

MAXIMUM_OF_N

The function MAXIMUM_Z_N returns the maximal value of variables specified in the list as parameters and is designed for using with a variable declared in the NONE type channel. The variables specified in the parameters list may be of various types, the function value is converted to FLOAT type. Neither the number of variables nor their order in the variable declaration, to which the MAXIMUM_OF_N function is assigned, are not limited. If the declaration does not fit in one line due to long list of variable names, it is necessary to separate it into several lines, which (except the last one) should end with continuation mark "\".

type of PLC variable - anyone
type of PC variable - FLOAT

parameters:

k, minimal number of the variables, that have been read correctly from a controller, for which the value returned by function is correct. The value of 0 means that all the variables have to be read correctly. If the number of variables that have been read correctly is lower than declared, the function returns a random value with error status.

Val_1,..Val_*n* names of ASMEN variables of any type

The function value is calculated as follows:
PCVal=max(Val_1,Val_2,..,Val_n)

MINIMUM_OF_N

The function MINIMUM_OF_N returns the minimal value of variables specified in the list as parameters and is designed for using with a variable declared in the NONE type channel. The variables specified in the parameters list may be of various types, the function value is converted to FLOAT type. Neither the number of variables nor their order in the variable declaration, to which the MINIMUM_OF_N function is assigned, are not limited. If the declaration does not fit in one line due to long list of variable names, it is necessary to separate it into several lines, which (except the last one) have to be ended with continuation mark "\".

type of PLC variable - any
type of PC variable - FLOAT

parameters:

k, minimal number of variables, that have been read correctly from a controller , for which the value returned by function is recognized as valid. The value of 0 means that all the variables have to be read correctly. If the number of variables that have been read correctly is lower than declared, the function returns a random value with error status.

Val_1,..Val_*n* names of ASMEN variables of any type

The function value is calculated as follows:
PCVal=min(Val_1,Val_2,..,Val_n)

FACTOR_FP

The function FACTOR_FP provides conversion of floating-point value of the process variable that has been read from a controller according to the formula:

*PCVal= A * PLCVal + B*

and conversion of value of the process variable that is to be sent to a controller according to the formula:

PLCVal = (PCVal - B) / A

type of PLC variable - FLOAT
type of PC variable - FLOAT

parameters:
A - FLOAT (default value 1.0)
B - FLOAT (default value 0.0)
#ABS - user receives an absolute value

NOTE *The function doesn't execute the operation of saving if it is used for the process variable of the NONE channel, for which the operation of saving with time and status wasn't declared.*

FACTOR_KG

The FACTOR_KG function enables conversion of the process variable value that has been read from a controller according to the formula:

*PCVal(IEEE 754) = A * PLCVal(KG) + B*

and conversion of the process variable value that is to be sent to a controller according to the formula:

PLCVal(KG) = (PCVal(IEEE 754) - B) / A

The FACTOR_KG conversion function provides conversion of the variable that has been read from a controller from the KG floating-point format to the floating-point format compatible with IEEE 754. When writing, conversion from the IEEE 754 compatible floating-point format to the KG floating-point format is performed.

type of PLC variable - FLOAT (KG)
type of PC variable - FLOAT

parameters:
A - FLOAT (default value 1.0)
B - FLOAT (default value 0.0)

NOTE *The function doesn't execute the operation of saving if it is used for the process variable of the NONE channel, for which the operation of saving with time and status wasn't declared.*

FACTOR_DW

The FACTOR_DW function enables conversion of 32-bit unsigned integer value of the process variable read from a controller according to the formula:

*PCVal = A * PLCVal + B*

and conversion of the process variable value that is to be sent to a controller according to the formula:

PLCVal = (PCVal – B) / A

type of PLC variable - DWORD
type of PC variable - FLOAT

parameters:
A - FLOAT (default value 1.0)
B - FLOAT (default value 0.0)

FACTOR

The FACTOR function provides conversion of the process variable value of WORD type (16-bit unsigned number) read from a controller according to the formula:

*PCVal = A * PLCVal + B*

and conversion of the process variable that is to be sent to a controller according to the formula:

PLCVal = (PCVal – B) / A

type of PLC variable - WORD
type of PC variable - FLOAT

parameters:
A - FLOAT (default value 1.0)
B - FLOAT (default value 0.0)

FACTOR_INT

The function FACTOR_INT provides conversion of the process variable value (16-bit signed number) read form a controller according to the formula:

*PCVal = A * PLCVal + B*

and conversion of the process variable that is to be sent to a controller according to the formula:

PLCVal = (PCVal – B) / A

type of PLC variable - INTEGER
type of PC variable - FLOAT

parameters:
A - FLOAT (default value 1.0)
B - FLOAT (default value 0.0)

FACTOR_LONG

The function FACTOR_LONG provides conversion of the process variable value (32-bit signed number) read from a controller according to the formula:

*PCVal = A * PLCVal + B*

and conversion of the process variable that is to be sent to a controller according to the formula:

PLCVal = (PCVal – B) / A

type of PLC variable - LONG
type of PC variable - FLOAT

parameters:
A - FLOAT (default value 1.0)
B - FLOAT (default value 0.0)

NEGBIT_BYTE

The function NEGBIT_BYTE transfers to the user an 8-bit value of the variable that has been read after inversion the bits corresponding to the mask.

type of PLC variable - BYTE
type of PC variable - BYTE

parameter:
8-bit mask in hexadecimal format

NEGBIT

The function NEGBIT transfers to the user a 16-bit value of the variable that has been read after inversion the bits corresponding to the mask.

type of PLC variable - WORD
type of PC variable - WORD

parameter:
16-bit mask in hexadecimal format

NEGBIT_DW

The function NEGBIT_DW transfers to the user a 32-bit value of the variable that has been read after inversion the bits corresponding to the mask.

type of PLC variable - DWORD
type of PC variable - DWORD

parameter:

32-bit in hexadecimal format

NOTHING

The NOTHING function transfers to the user a 16-bit value of the process variable in the form that has been read by a driver without performing any additional conversions.

type of PLC variable - WORD
type of PC variable - WORD

parameters:

MIN	- number of the same type as the PC variable, it has no influence on value conversion, is used to define the range of changes of processed variable;
MAX	- number of the same type as the PC variable, it has no influence on value conversion, it used to define the range of changes of processed variable.

NOTHING_BYTE

The function NOTHING_BYTE transfers to the user a value of the process variable in 8-bit number format in the form that has been read by driver without executing any additional conversions.

type of PLC variable - BYTE
type of PC variable - BYTE

parameters:

MIN	- number of the same type as the PC variable, it has no influence on value conversion, it used to define the range of changes of processed variable;
MAX	- number of the same type as the PC variable, it has no influence on value conversion, it used to define the range of changes of processed variable.

NOTHING_DD

The function NOTHING_DD transfers to the user a value of the process variable represented in a controller by double word, exchanging places of higher and lower word.

type of PLC variable - DWORD
type of PC variable - DWORD

parameters:

MIN	- number of the same type as the PC variable, it has no influence on value conversion, it used to define the range of changes of processed variable;
MAX	- number of the same type as the PC variable, it has no influence on value conversion, it used to define the range of changes of processed variable.

NOTHING_DW

The function NOTHING_DW transfers to the user a 32-bit value of the process variable in the format that has been read by driver without executing any additional conversions.

type of PLC variable - DWORD
type of PC variable - DWORD

parameters:

MIN	- number of the same type as the PC variable, it has no influence on value conversion, it used to define the range of changes of processed variable;
MAX	- number of the same type as the PC variable, it has no influence on value conversion, it used to define the range of changes of processed variable.

NOTHING_FP

The function NOTHING_FP transfers to the user a value of the process variable of FLOAT type in the form transferred by a driver without performing any additional conversions.

type of PLC variable - FLOAT
type of PC variable - FLOAT

parameters:

MIN	- number of the same type as the PC variable, it has no influence on value conversion, it used to define the range of changes of processed variable;
MAX	- number of the same type as the PC variable, it has no influence on value conversion, it used to define the range of changes of processed variable.

NOTHING_INT

The function NOTHING_INT transfers to the user a value of the process variable in 16-bit signed number format in the form read by a driver without executing any additional conversions.

type of PLC variable - INTEGER
type of PC variable - INTEGER

parameters:

MIN	- number of the same type as the PC variable, it has no influence on value conversion, it used to define the range of changes of processed variable;
MAX	- number of the same type as the PC variable, it has no influence on value conversion, it used to define the range of changes of processed variable.

NOTHING_LONG

The function NOTHING_LONG transfers to the user a value of the process variable in 32-bit signed number format in the form read by a driver without performing any additional conversions.

type of PLC variable - LONG
type of PC variable - LONG

parameters:

MIN	- number of the same type as the PC variable, it has no influence on value conversion, it used to define the range of changes of processed variable;
MAX	- number of the same type as the PC variable, it has no influence on value conversion, it used to define the range of changes of processed variable.

NOTHING_KG

The function NOTHING_KG transfer to the user a value of the process variable that have been read from a controller in the KG floating-point format (a specific format of SIEMENS controllers) and changed to the floating-point format compatible to IEEE 754. When writing, conversion from the IEEE 754 compatible floating-point format to the KG floating-point format is executed.

type of PLC variable - FLOAT (KG)
type of PC variable - FLOAT

parameters:

MIN	- number of the same type as the PC variable, it has no influence on value conversion, it used to define the range of changes of processed variable;
MAX	- number of the same type as the PC variable, it has no influence on value conversion, it used to define the range of changes of processed variable.

NOTHING_TEXT

The function NOTHING_TEXT transfers to the user a value of the process variable in form of an ASCII string ending with zero character.

type of PLC variable - BYTE
type of PC variable - BYTE

it has no parameters

ON/OFF

The function ON/OFF has been developed for test purposes, when tests are carried out without controller (using NONE protocol). It enables the user to generate binary values.

type of PLC variable - WORD

type of PC variable - WORD

parameters:
ON value - WORD (1 by default)
OFF value - WORD (0 by default)
initial value - WORD (0 by default)

SAW

The function SAW has been developed for test purposes, when tests are carried out without controller (using NONE protocol). It enables saw-shape trends generation with specified limit and step values. The process variable is of the INTEGER type.

type of PLC variable - INTEGER
type of PC variable - INTEGER

parameters:
initial increment /decrement trend (1 - up, 0 - down) - INTEGER (default value - 1)
step - INTEGER (default value - 1)
maximum - INTEGER (default value - 100)
minimum - INTEGER (default value - 0)
initial value - INTEGER (default value - 0)

SAW_FP

The function SAW_FP has been developed for test purposes, when tests are carried out without controller (using the NONE protocol). It enables saw-shape trends generation with specified limit and step values. The process variable is of FLOAT type.

type of PLC variable - FLOAT
type of PC variable - FLOAT

parameters:
initial increment /decrement trend (1 - up, 0 - down) - INTEGER (default value - 1)
step - FLOAT (default value - 1.0)
maximum - FLOAT (default value - 100.0)
minimum - FLOAT (default value - 0.0)
initial value - FLOAT (default value - 0.0)

PERCENT

The function PERCENT provides conversion of value of the *INTEGER* process variable read from a controller to the PC's *FLOAT* format value according to the formula:

*PCVal = (PLCVal / k) * 100.0*

and conversion of value of the process variable that is to be sent to a controller from *FLOAT* format to *INTEGER* format according to the formula:

*PLCVal = (PCVal / 100.0) * k*

where ***k*** is the function parameter.

type of PLC variable - INTEGER
type of PC variable - FLOAT

parameters:
Reference value - FLOAT (default value - 2048.0)

PERCENT_FP

The function PERCENT_FP provides conversion of value of the process variable read from a controller to the value according to the formula:

*PCVal = (PLCVal / k) * 100.0*

and conversion of value of the process variable that is to be sent to a controller according to the formula:

*PLCVal = (PCVal / 100.0) * k*

where ***k*** is the function parameter.

type of PLC variable - FLOAT
type of PC variable - FLOAT

parameters:
Reference value - FLOAT (default value - 2048.0)

SHIFT_L

The function SHIFT_L reads a variable value from a controller, shifts it left by the number of bits given in the first parameter and then performs logical AND of the shifted value and the second parameter.

While value of the system variable is sent to a controller it's left unchanged.

type of PLC variable - WORD
type of PC variable - WORD

parameter:
number of bits to shift by in the range 0 to 16 (0 by default)
16-bit hexadecimal mask value (FFFF by default)

SHIFT_R

The function SHIFT_R reads a variable value from the controller, shifts it right by the number of bits given in the first parameter and then performs logical AND of the shifted value and the second parameter.

While value of the system variable is sent to a controller it's left unchanged.

type of PLC variable - WORD
type of PC variable - WORD

parameter:
number of bits to shift by in the range 0 to 16 (0 by default)
16-bit hexadecimal mask value (FFFF by default)

AVERAGE

The function AVERAGE calculates an average from samples of process variable values that have been read from a controller according to the formula:

*PCVal = MinPCVal + (PLCVal - MinPLCVal) * k*

and provides conversion of the process variable that is to be sent to a controller according to the formula:

*PLCVal = MinPLCVal + (PCVal - MinPCVal) * k*

The *k* factor is calculated as follows:

k = (MaxPCVal - MinPCVal)/(MaxPLCVal - MinPLCVal)

The average is calculated for *Period* time period (given in minutes), and the result is regarded as correct if the percentage rate of correctly read samples is equal *Rate*.

The function parameter may be ONLY a number (expressed in minutes), which meets one of the following conditions:
divides an hour into equal periods
divides 24 hours into equal periods being a multiple of an hour.

Calls of ASPAD must be synchronized with the system clock but not with the moment of the **asix** start-up.

type of PLC variable - anyone
type of PC variable - FLOAT

parameters:
MinPLCVal - no reference
MaxPLCVal - no reference
MinPCVal - FLOAT (default value 0.0)
MaxPCVal - FLOAT (default value 2048.0)
Period - INTEGER (default value 5)
Rate - INTEGER (default value 80)

AVERAGE_OF_N

The function AVERAGE_OF_N returns an actual value of arithmetic average of the variables specified in the list as parameters and is designed for using with a variable declared in the NONE type channel. The variables specified in the parameters list may be of various types, the function value is converted to FLOAT type. Neither the number of variables nor their order in the variable declaration, to which the AVERAGE_OF_N function is assigned, are not limited. If the declaration does not fit in one line due to long list of variable names, it is necessary to separate it into several lines, which (beside the last one) have to be ended by continuation mark "\".

type of PLC variable - anyone
type of PC variable - FLOAT

parameters:
k, minimal number of variables, that have been read correctly from a controller, for which the result of function evaluation is valid. The value of 0 means that all the variables have to be read correctly. If the number of variables that have been read correctly is lower than declared, the function returns a random value with error status.
Val_1,..Val_*n* names of ASMEN variables of any type

The function value is calculated as follows:

PCVal=(Val_1+Val_2+..+Val_n)/n

CONSTANT

If the function parameter is an INTEGER type number then the function assignes the value of the number to a process variable.

If the function parameter is a process variable name then the function assignes the actual value of the variable-parameter to the process variable. The real type of the variable-parameter must be INTEGER.

The function is used for reading only.

type of PC variable - INTEGER
parameters:
IntegerVal - INTEGER (default value 0)
or
Name - STRING

CONSTANT_FP

If the function parameter is an FLOAT type number then the function assignes the value of the number to a process variable.

If the function parameter is a process variable name then the function assignes the actual value of the variable-parameter to the process variable. The real type of the variable-parameter must be FLOAT.

The function is used for reading only.

type of PC variable - FLOAT

parameters:
FloatVal - FLOAT (default value 0)
or
Name - STRING

STATUS

The function returns the current status of the variable that is the function parameter.

The variable that uses the function has to be defined in the NONE or NETSRV channel.

The function allows only a reading operation.

EXAMPLE

VAL, variable from the controller, CHANNEL_PLC,1,1,NIC
STAT, variable showing the VAL1 variable status,CHANNEL_NONE,1,1,STATUS,VAL

☑ *SUM_OF_N*

The function SUM_OF_N returns an actual value of the sum of the variables specified in the list as parameters and is designed for using with a variable declared in the NONE type channel. The variables specified in the parameters list may be of various types, the function value is converted to FLOAT type. Neither the number of variables nor their order in the variable declaration, to which SUM_OF_N function is assigned, are not limited. If a declaration does not fit in one line due to long list of variable names so it is necessary to separate it into several lines, which (beside the last one) have to be ended by continuation mark "\".

type of PLC variable - anyone
type of PC variable - FLOAT

parameters:

k, minimal number of variables, that have been read correctly from a controller, at which the correctness of function evaluation is recognized. The value of 0 means that all the variables must have been read correctly. If the number of variables that have been read correctly is lower than declared, the function returns a random value with error status.

Val_1,..Val_*n* names of ASMEN variables of any type

The function value is calculated as follows:

PCVal=Val_1+Val_2+..+Val_n

☑ *SLIDER*

The function SLIDER resets to zero the MSB when a value of the process variable is read from a controller and sets to 1 in MSB when a value of the variable is written to a controller.

type of PLC variable - WORD
type of PC variable - WORD

It has no parameters.

NOTICE *The function doesn't execute the operation of saving if it is used for the process variable of the NONE channel, for which the operation of saving with time and status wasn't declared.*

☑ *SLIDER1*

The function SLIDER1 works like the function ANALOG with the following differences:

- when reading from a controller the MSB of process variable value is reset to 0,
- before writing to a controller the MSB of process variable value is set to 1.

type of PLC variable - WORD
type of PC variable - WORD

parameters (as for ANALOG function):
MinPLCVal - INTEGER (default value - 0)
MaxPLCVal - INTEGER (default value - 2048)
MinPCVal - INTEGER (default value - 0)
MaxPCVal - INTEGER (default value - 2048)

NOTICE *The function doesn't execute the operation of saving if it is used for the process variable of the NONE channel, for which the operation of saving with time and status wasn't declared.*

☑ *SLIDER1_FP*

The function SLIDER1_FP works like the function ANALOG_FP with following differences:

- when reading from a controller the MSB of process variable value is reset to 0,
- before writing to a controller the MSB of process variable value is set to 1.

type of PLC variable - WORD
type of PC variable - FLOAT

parameters (as for the function ANALOG_FP):
MinPLCVal - INTEGER (default value 0)
MaxPLCVal - INTEGER (default value 2048)
MinPCVal - FLOAT (default value 0.0)
MaxPCVal - FLOAT (default value 2048.0)

NOTICE *The SLIDER1_FP function operates like NOTHING_FP function when it is used for variables of the NONE channel.*

☑ *TABLE*

The function TABLE converts the value of a source variable according to the nonlinear function described in the table transferred in the text file, each line of which consists of the variable value and of the real value assigned to it. The lines beginning from the character **#** or **;** are ignored (they may include a comment). Intermediate values are the subject to linearization, if a source value exceeds the range declared in declaration file, the function returns a measuring error. The variable, to which the function TABLE is assigned, must be placed in the NONE type channel.

It is possible to declare in the ASMEN section a path to the directory where all the tables, used by conversion functions of a given application, will be stored – the following item is used for this purpose:

PATH_TABLE=<path>

The function returns the value

PCVal=f(PCVal_1)

where:
PCVAL_1 - name of previously declared the ASMEN variable;
f(x) - function that is defined in table form in a text file.

type of PC variable - FLOAT

parameters:
PCVAL_1 - FLOAT|LONG|DWORD|WORD|INT16
definition_file - text

EXAMPLE

An examplary variable declaration:

AAA, Description – variable AAA depending on BBB, , NONE, 1, 1, TABLE,BBB,tablica1.txt

An example of tablica1.txt file defining the table:

10 15.7
20 21
30 26.4
40 32
50 37.1

As a separator it is possible to use *Space* or *Tab*. For fractional numbers you should use dot (.) as a decimal separator.

WAVERAGE_OF_N

The function WAVERAGE_OF_N returns an actual value of weighted average of the variables specified in the list as parameters together with their weights. The parameters are used in pairs: *variable_weight/variable_value*. An additional parameter, which precedes the list of weights and variables, is a number defining the minimum number of pairs *weight/variable*, which have to be read correctly in order that the function result is valid. The variables specified in the list may be of various types. The function value is converted to FLOAT type. Neither the number of variables nor their order in the variable declaration, to which the WAVERAGE_OF_N function is assigned, are not limited. The successive variables are separated with space characters. If the declaration does not fit in one line due to long list of variable names, it is necessary to split it into several lines, which (except the last one) has to be ended by continuation mark "\".

type of PLC variable - any
type of PC variable - FLOAT

parameters:

k, minimal number of pairs weight/variable that have been read correctly from a controller, at which the correctness of function evaluation is recognized. The value of 0 means that all the variables must have been read correctly. If the number of variables that have been read correctly is lower than that has been declared, the function returns a random value with error status.

weight_1,Val_1,..weight_n,Val_n names of ASMEN variables of any type

The function value is calculated as follows:

PCVal=(weight_1*Val_1+weight_2*Val_2+..+weight_n*Val_n)/(weight_1+weight_2+…+weight_n)

EXAMPLE

An examplary declaration of the ASMEN variable:
WEIGHTED_AVERAGE, I0.2, CHANNEL1, 1, 1, WAVERAGE_OF_N, 0, WEIGHT_1 VAL_1 WEIGHT_2 VAL_2 WEIGHT_3 VAL_3

DATA_SCOPE

The function DATA_SCOPE provides conversion of process variable read from a controller according to the formula:

PCVal = MinPLCVal + (PLCVal – MinPCVal) * k

and conversion of the process variable value that is to be sent to a controller according to the formula:

PLCVal = MinPCVal + (PCVal – MinPLCVal) / k

where:

k = (MaxPLCVal – MinPLCVal)/(MaxPCVal – MinPCVal)

PLCVal - type of PLC variable

PCVal - type of PC variable

type of PLC variable -FLOAT
type of PC variable -FLOAT

parameters:

MinPCVal -FLOAT
MaxPCVal -FLOAT
MinPLCVal -FLOAT
MaxPLCVal -FLOAT

When using the NONE channel the type of variable is set on FLOAT type.

6.6. Diagnostics of Correct Format of Numbers Read from Controllers

The ASMEN module checks if FLOAT numbers read from controllers are compatible to the IEEE 754 format.

If a number of value NAN or INF is found, then the value of the variable (assigned to this number) is set to 0 and its error status is set to AVD_BAD. Simultaneously, in the window of '*Control Panel*' the message containing error type (NAN or INF) and the name of the variable with bad format is displayed. If a log file is declared for a given application, then the message is written there too.

6.7. Drivers and Transmission Protocols

asix system includes a set of drivers that handle the following types of data transfer with controllers of an industrial process.

Driver	Protocol
ADAM	- protocol for ADAM-4000 modules of ADVANTECH
AGGREGATE	- the driver allows definition of variables, values of which are generated as a result of calculations performed on other variables of the **asix** system (source variables)
CtAK	- the AK protocol allows data exchange between **asix** system computers and Emerson MLT2 analyzers
AM_SA85	- protocol for communication with the Modbus Plus network of Schneider Automation
AREVA	- allows to exchange data between asix and digital protection devices MiCOM of AREVA; the list of serviced devices includes MiCOM P127 and MiCOM P34x series
AS511	- protocol using programmer interface of SIMATIC PLCs of SIEMENS
AS512	- protocol of CP524/525 communication processors for SIMATIC PLCs of SIEMENS
AS512S7	- protocol of CP340 communication processors for SIMATIC S7 PLCs of SIEMENS
BAZA	- the driver enables data import from databases to the **asix** system
BUFOR	- general purpose protocol for information exchange with user programs by means of a shared memory
CALEC MCP	- Calec MCP driver retrieving the current values of variables from CALEC MCP devices of Aquametro according to the protocol described in the document „MCP Datenauslesung mit dem lowlevel Protokoll"
CAN_AC_PCI	- protocol for data exchange between SELECONTROL MAS PLCs of Selectron Lyss AG and **asix** system computers

CANOPEN	- CANOPEN network protocol of SELECTRON MAS PLCs of Selectron Lyss AG
COMLI	- protocol (COMunication Link) for communication with ABB SattCon, AC 800C, AC 800M, AC 250 PLCs; data are transferred via RS-232 or RS-485 serial interfaces
CZAZ	- allows to exchange data between **asix** system and digital protection devices CZAZ-U and CZAZ-UM of ZEG-Energetyka
DATAPAF	- protocol for connection with DataPAF energy counters
DDE	- driver defining a channel of the ASMEN module referring to variables shared by a DDE server
DP	- protocol for devices compatible with PROFIBUS DP by using a PROFIboard card
DP5412	- protocol for devices compatible with PROFIBUS DP by using Siemens cards
DMS285	- protocol for emission meter D-MS285 computers
DMS500	- protocol for emission meter D-MS500 computers (previous name – DURAG)
DSC	- protocol for data exchange between **asix** system computers and DSC 2000 controllers
DXF351	- protocol for communication with Compart DXF351 devices of Endress+Hauser
CtEcoMUZ	- protocol for data exchange between the **asix** system and Microprocessor Protecting ecoMUZ Devices made by JM Tronik
FESTO	- protocol using a diagnostic interface for FESTO PLCs
FILE2ASIX	- the driver enables data import from text files to the **asix** system
FP1001	- protocol for water and steam flow monitors of METRONIC Kraków
GFCAN	- protocol of CAN network with use of communication card of Garz & Fricke Industrieautomation GmbH
K3N	- protocol for data exchange between K3N meters family of OMRON and **asix** system computers
K-BUS	- protocol K-BUS used for data exchange between VIESSMANN Dekamatic boilers controllers connected to a Dekatel-G (or Vitocom 200) concentrator and **asix** system computers
CtLG	- protocol of LG Master-K and Glofa GM PLCs
CtLogo	- driver of Logo OBA5 controllers from SIEMENS
LUMBUS	- protocol for data exchange between RG72 controllers manufactured by Lubuskie Zakłady Aparatów Elektrycznych (Electrical Measuring Instrument Works) "LUMEL" in Zielona Góra and **asix** system computers
CtM200	- for data exchange between the asix system and the M210G flow computer from Spirax Sarco
MACMAT	- GAZ-MODEM protocol used for communication with MACMAT stations
M-BUS	- subset of standard protocol for data reading from measuring devices used by MULTICAL heat meters of KAMSTRUP A/S
MEC	- protocol of data exchange between **asix** system and MEC07 and MEC08 heat meters manufactured by Instytut Techniki Cieplnej (Institute of Thermal

	Technology) in Lodz. Data are transferred with use of a standard RS-232 interface.
MELSECA	- protocol of A1SJ71C24-R2 communication processor for MELSEC-A PLCs
MEVAS	- protocol for data exchange between the MEVAS emission meter computer produced by Lubuskie Zakłady Aparatów Elektrycznych (Electrical Measuring Instrument Works) "LUMEL" in Zielona Góra and **asix** system computers
MicroSmart	- driver for data exchange with MicroSmart controllers from IDEC
MODBUS	- subset of standard communication protocol used by AEG Modicon GE Fanuc PLCs
MODBUS_TCPIP	- protocol of data exchange between **asix** system and computers/devices by means of the MODBUS protocol on the basis of Ethernet with the TCP/IP protocol
MODBUSSLV	- MODBUS protocol, in which **asix** operates as SLAVE
MPI	- protocol of MPI interface of SIMATIC S7 PLCs of SIEMENS; a serial interface
MPS	- serial interface protocol for MPS measuring gauges of a power network from OBR Metrologii Elektrycznej in Zielona Góra
MSP1X	- protocol MSP1X used for data exchange between MSP1X PLCs of ELMONTEX and **asix** system computers
MultiMuz	- used to exchange data between the asix system and te MultiMUZ microprocessor-based security devices manufactured by Warsaw-based JM-Tronik
MUPASZ	- hollow (virtual) channel protocol
CtMus04	- allows data between the asix system and the microprocessor-based control devices MUS-04 manufactured by ELEKTROMETAL S.A. from Cieszyn to be exchanged
MUZ	- protocol for data exchange between Microprocessor Security Devices of MUZ-RO type;
CtNCP	- is used to exchange data between the asix system and MN-series controllers from Invensys (former Satchwell)
NONE	- NONE protocol enables: - **asix** application testing in simulation mode, - data exchange between **asix** programs by means of process variables;
NetLink	- is used for data exchange between asix computers and SIMATIC S7 PLCs by using an MPI/Profibus bus and a NetLink Lite SYSTEME HELMHOLZ module
NetLinkPro	- to communicate with the S7 controllers via the NETLink PRO gateway
OMRON	- enables data exchange between OMRON PLCs and **asix** system computers
OPC	- the driver defining a channel of ASMEN module retrieving the variables shared by an OPC server
CtPmc4000	- is used for data exchange between the asix system and POLON 4800 fire protection station according to protocol PMC-4000
PPI	- protocol for SIEMENS S7-200 PLCs

PROTHERM	- driver for data exchange between Protherm 300 DIFF PLCs of Process-Electronic GmbH and **asix** system computers
PROTRONICPS	- PROTRONIC PS protocol of Hartmann & Braun
S700	- protocol S700 used for data exchange between Maihak 3700 gas analyzers and **asix** system computers
S7_TCPIP	- is used for data exchange with SIMATIC S7-series controllers through the Ethernet connection with the use of a standard computer network card
SAPIS7	- protocol of SIMATIC S7 PLCs with use of the MPI interface or PROFIBUS communication processor (an implementation of S7 function)
S-BUS	- S-BUS protocol used for data exchange between PCD PLCs of SAIA Burgess Electronics and **asix** system computers
CtSbusTcpip	- driver for data exchange between family of PCD SAIA-Burgess PLCs and **asix** system computers
SINECH1	- protocol of CP1430 communication processors of SIMATIC S5 PLCs (Ethernet)
SINECL2	- protocol of CP5430 communication processors of SIMATIC S5 PLCs of SIEMENS
CtSi400	- is used for data exchange between asix computers and a Sintony Si 400 alarm central of SIEMENS
SNPX	- driver for data exchange between **asix** system computers and GE Fanuc 90-30 PLCs as well as GE Fanuc 90 CMM and PCM modules
SPA	- protocol used for communication with devices connected to SPA bus of the ABB company
SRTP	- driver used for data exchange between the **asix** system and GE Fanuc Automation VersaMax Nano/Micro PPLCs using an IC200SET001 converter and WersaMax 90 PLCs using the IC693CMM321 communication module; via Ethernet with the TCP/IP protocol
TALAS	- protocol of TALAS emission computers
CtTwinCAT	- driver for data exchange between the **asix** system and the TwinCAT system of Beckhoff Industrie Elektronik
ZDARZENIE ZMIENNA	- driver for generating process variables of WORD type (16-bit word) on the basis of actual values of alarm events in the **asix** system
CtZxD400	- driver of protocol of electric energy counters of ZxD400 type manufactured by Landys & Gyr

The package of available protocols will be systematically expanded. The ASKOM company is ready to develop on customer request any transmission protocol according to the rules defined in the price list of the **asix** system.

See description of drivers and their protocols in:
Communication Drivers – User's Manual (CommunicationDrivers.pdf)

6.8. Data Exchange in Local Area Network

The **asix** system can make process data available also on stations which have no direct connections to the process controllers. The local area network (LAN) is used for this purpose. The stations, connected directly to controllers, operate as ***data servers*** and make data available for other stations (***remote stations***) by means of a LAN.

asix transmission channels used for data exchange via a LAN will be named as *network channels*.

Data exchange in a LAN between stations of the **asix** system is executed by the NETSRV module. This module is loaded automatically while **asix** installation.

The network configuration is made in the XML application configuration file with use of Architect module.

See: Architect user's manual, chapter 3.7. *Configuration of network server related current data parameters*

6.8.1. Declaration of Network Channel

Declaration of network channel needs the following parameters to be set up:

Name – logical name of the network channel;
Driver – name of the NETWORK driver;
Server name – name of the data server computer (optional).

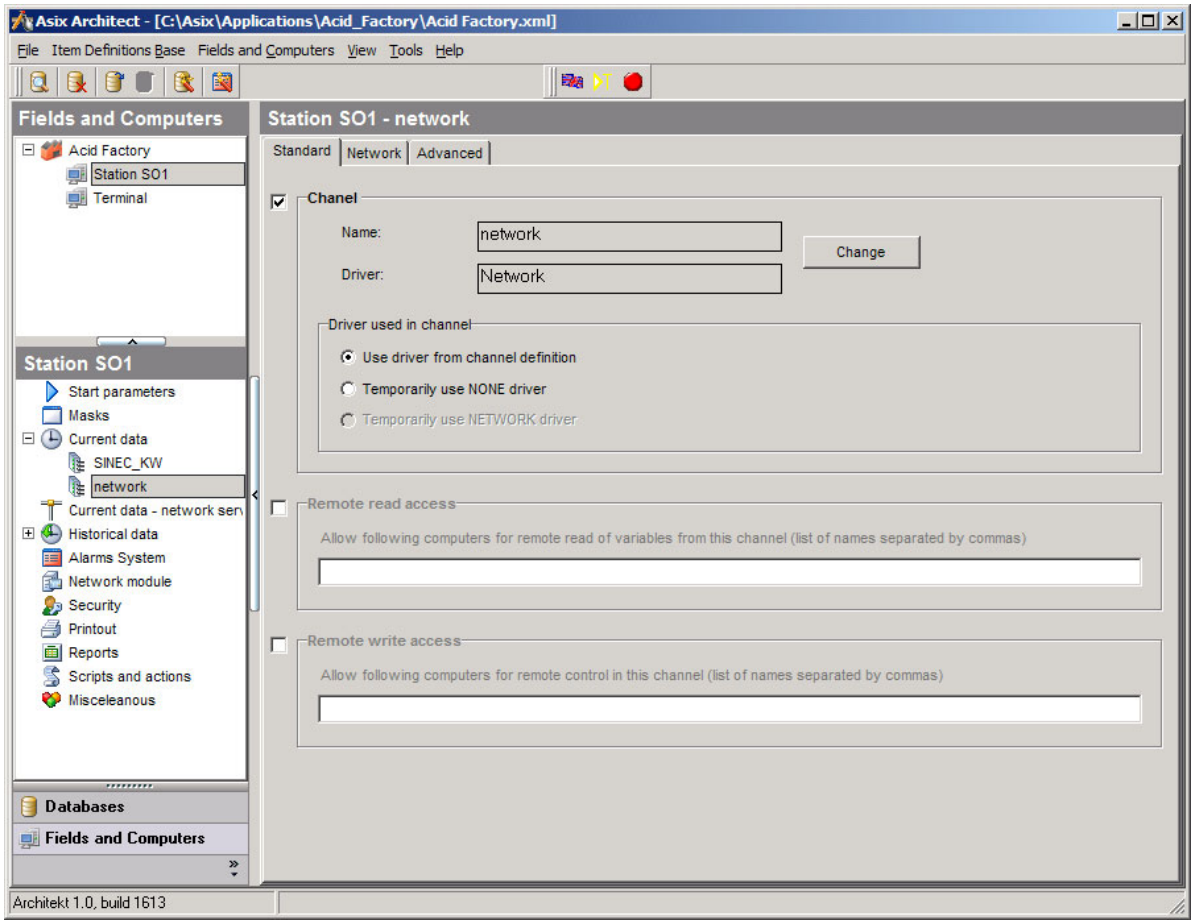


Figure 6-3. Declaration of Network Channel – Obligatory Parameters.

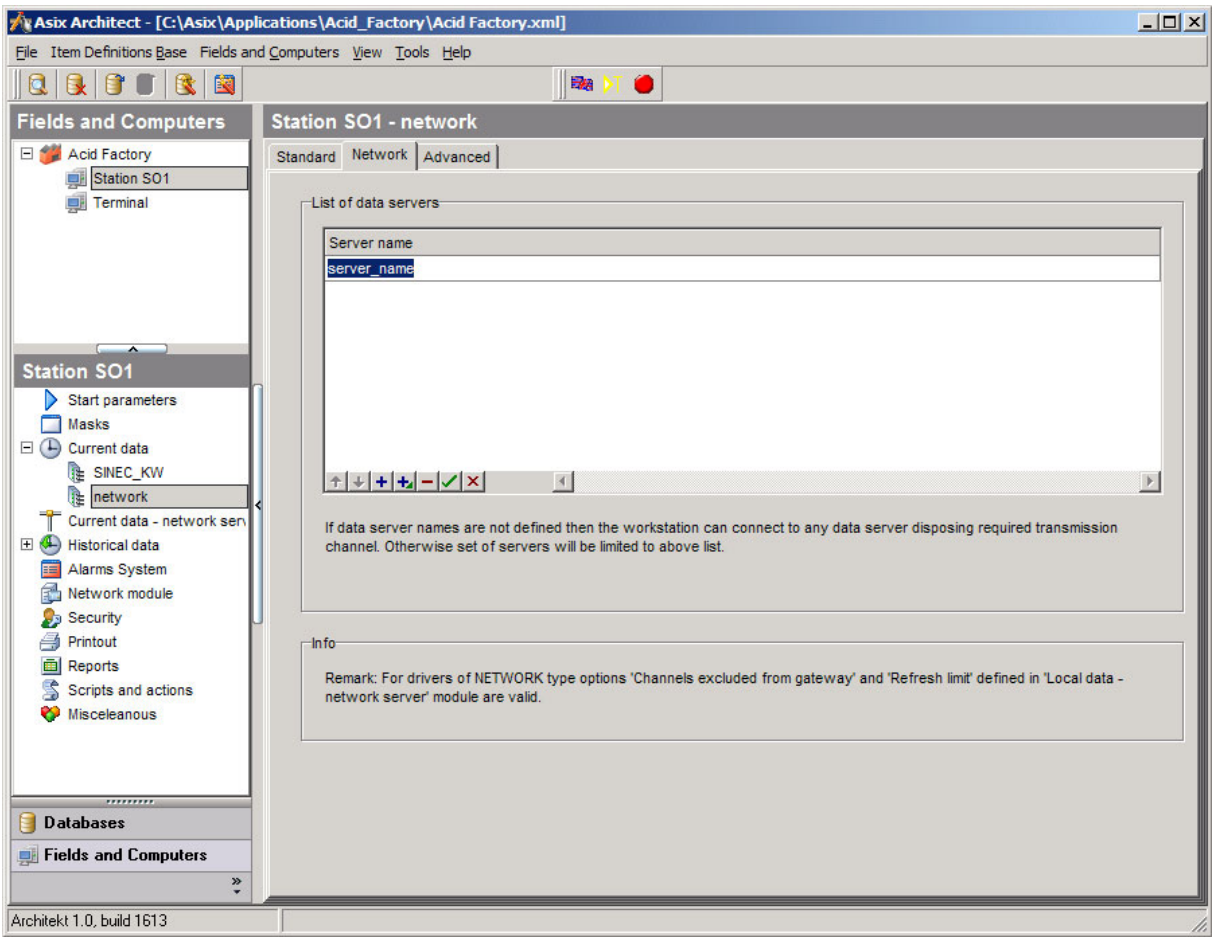


Figure 6-4. Declaration of Network Channel - Optional Parameter.

If such a declaration is given, then all process variables declared in the *name* channel will be read from the data server, on which the transmission channel of the same logical name is declared. The *name* channel of the data server must meet one of the following conditions:

- be directly connected to the controller,
- be a NONE channel,
- work in the GATEWAY mode (see: *6.8.3. GATEWAY Station Operating-Mode*)

If names of data servers are not given in the network channel declaration then the work station may be connected to any data server having at its disposal a required transmission channel. Otherwise the range of servers will be limited to these computers, the names of which were placed in the network channel definition.

6.8.2. Process Variable Declarations in Network Protocol

The remote station can access a process variable on data server under the following conditions:

- the same process variable must be declared on the remote station and on the data server,
- the declarations of the process variable must be identical on both sides.

If any of the above conditions is not fulfilled, then the *'Operator Panel'* message about the wrong variable declaration will be displayed while the connection will be established.

6.8.3. GATEWAY Station Operating-Mode

In the normal mode the **asix** data server makes accessible only the data of physical channels (directly connected to process devices and controllers). However there are situations when the station is to be both a server and a client for a given data. It's typical when the **asix** station works as a gateway between two separate networks. In such a situation the network channels of the station should be treated in the same way as physical channels, taking in consideration the data propagation in the network. Such station work-mode is named GATEWAY.

The gateway operating-mode should be declared by setting up the *Gateway mode* parameter:

Architect > *Fields and Computers* > *Current data – network server* > *Standard* tab

There are all options for operation of the netsrv.dll program, which realizes the **asix** operation in gateway mode.

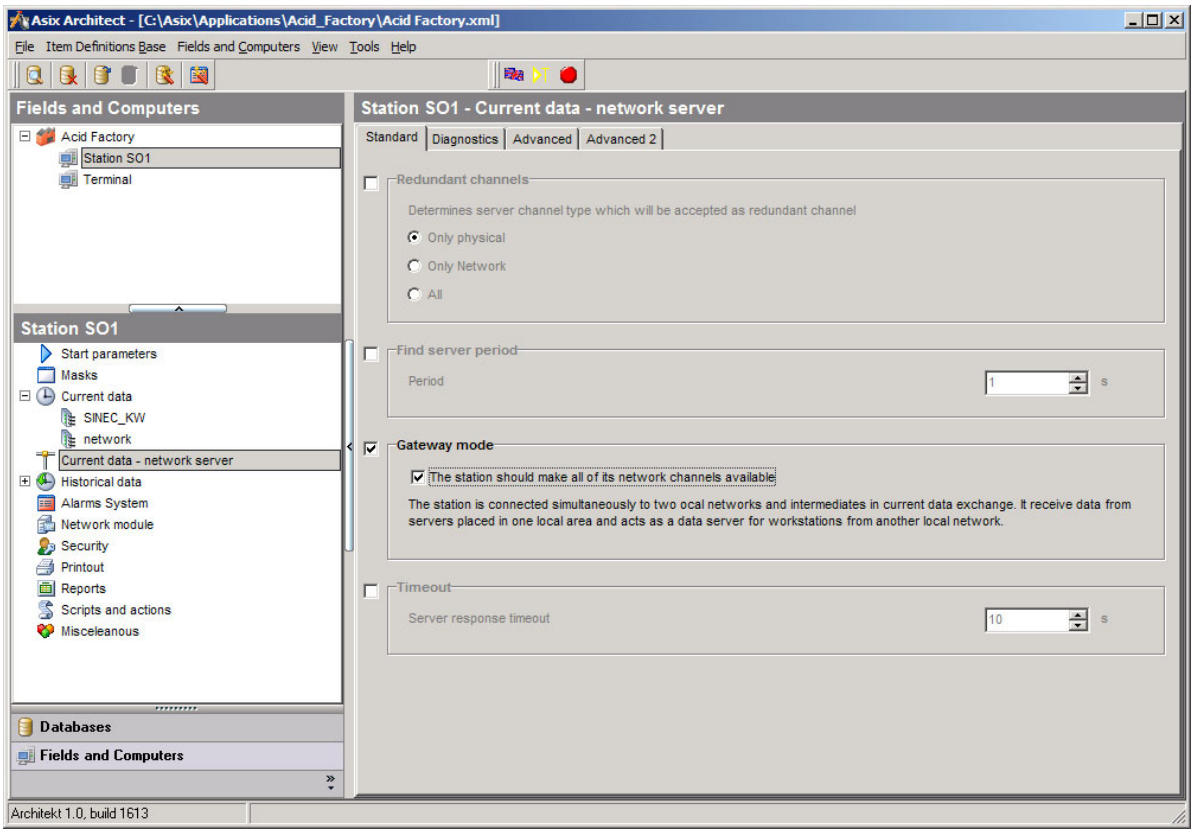


Figure 6-5. GATEWAY Station Operating-Mode Declaration.

6.8.4. Redundancy

The **asix** system is provided with features of fault-tolerant systems. One of them is the transmission channel redundancy. If the physical channel on a given station is broken, then the data will still be available via the network channel (if there is another station in the network connected directly to the same device).

Redundancy can be declared only for physical channels (the ones directly connected to the controller or device).

The declaration of redundant channel should be declared with use of the Redundancy option placed in:

Architect > *Fields and Computers* > *Current data – network server / Network channel* mode > *Advanced tab*

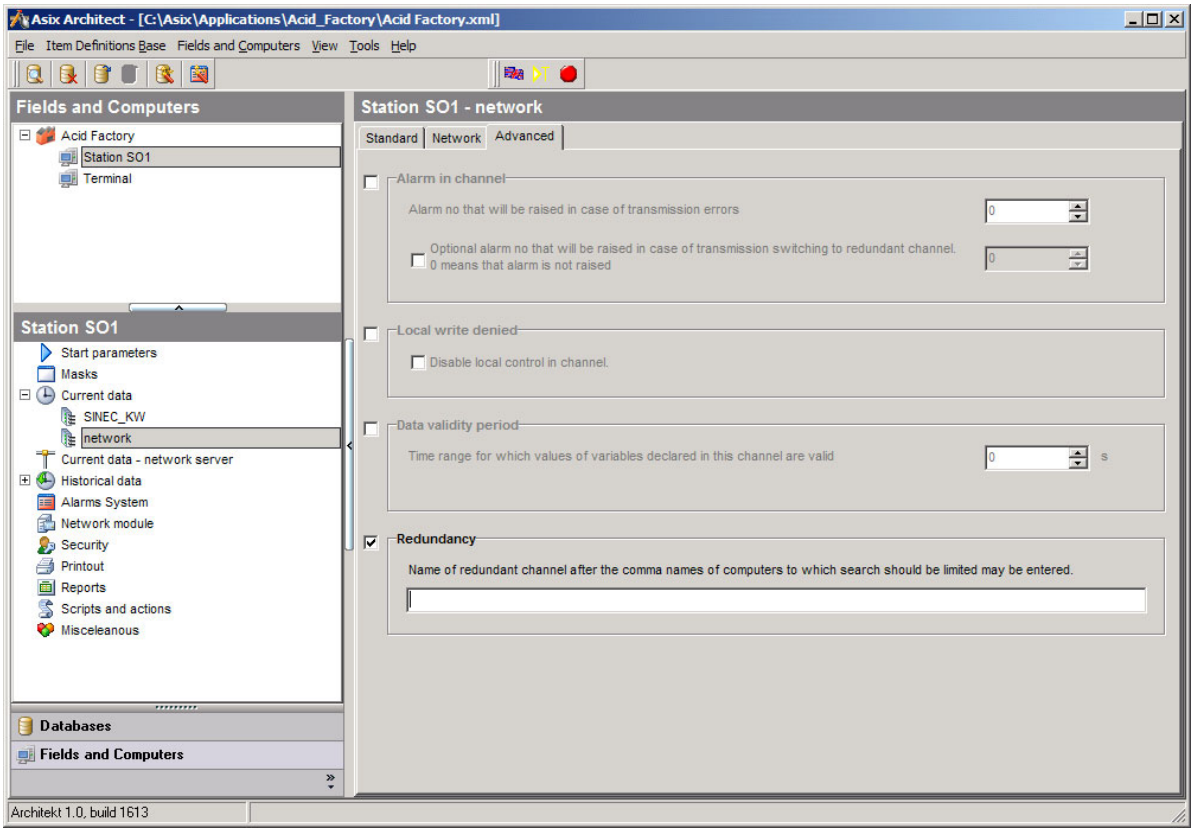


Figure 6-6. Declaration of Redundancy Channels.

The redundant channel declaration has the following syntax:

REDUNDANCY=*logical_name* [**,** *server_1***,** *server_2***, ...,** *server_n*]

where:

logical_name	- logical name of the earlier declared physical channel;
REDUNDANCY	- the constant part of declaration;
server_i	- the name of the **asix** computer which can be used as a redundant data server (optional).

Such declaration means that if the *logical_name* physical channel is broken, then data will be read from another data server of the same channel.

If in the redundant channel declaration, the names of data servers are not given, then an alternate channel may be accessed by any data server having at its disposal a required transmission channel. Otherwise an alternate channel will be searched only on these data servers, the names of which were placed in the redundant channel declaration.

On the **asix** station st1, via which the controls from an other **asix** station st2 are executed, a permission to carry out these controls should be declared:

Architect > *Fields and Computers* > *Current data – network server / Network channel* mode > *Standard tab* > *Remote write access* option

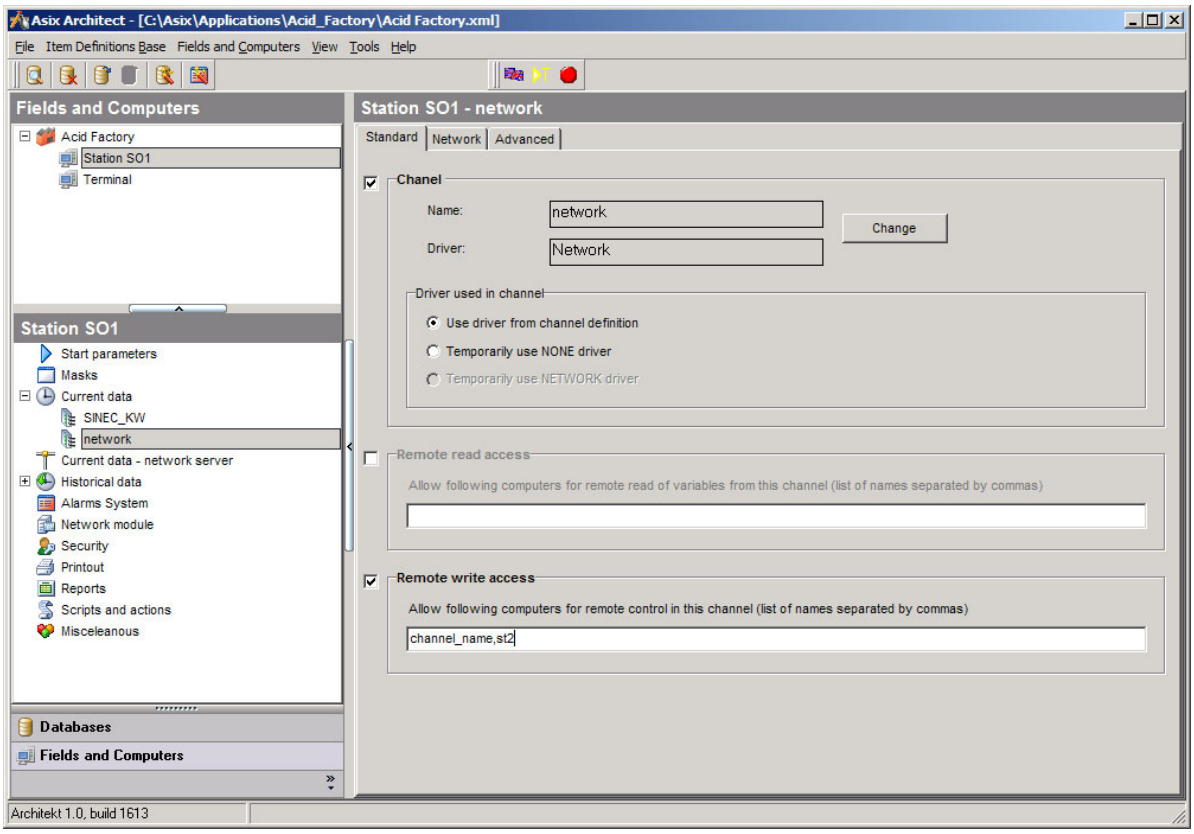


Figure 6-7. Declaration of Remote Control Execution.

7. ASPAD - Data Archiving Module

ASPAD is an **asix** module designed for recording and later using of time trends of variables. In **asix** system those trends are used for creating charts and reports, and can also be used by scripts and calculator objects of derived variables. They can also be exported with use of the **AsixConnect** program in DDE or OLE format to other programs, e.g. EXCEL.

Below description regards to ASPAD program version 6, which is significantly different from its earlier versions.

7.1.1. Basic Features of ASPAD Module

Version 6 of the ASPAD program is provided with the following features.

- Archiving variables of any type allowed in the ASMEN module, under condition that they are scalar variables. They can be either process variables, or obtained in result of calculations or simulations. Current version doesn't allow array archiving.
- Besides the value and the time it also stores the value status compatible with the OPC standard.
- Collects the data either in files with specifically designed structure (D, M, Y archive with separate day, month or year files), or in typical databases (B archive).
- For B-type it allows obtaining history data from devices provided with memory.
- For B-type it allows archiving data retrieved asynchronously and non-sequentially, for example from telemetry systems.
- Allows construction of more or less complex conditions of specific variables archiving.
- For D, M and Y types saves disk memory, saving only significant changes of archived values and using simple methods of compression.
- Due to proper internal file structure allows fast access to data from any archiving period.
- Allows splitting the archive into day, month or year files.
- Allows declaring D, M and Y-type files storing time, after which the old files would automatically be deleted (and through the period of 10 % of the specified time their backup copies are stored to avoid problems in case of accidental changes of computer's time).
- Uses filenames convention, with resource name attached and readable time encoding till year 9999.
- Can actively archive or only restore old data.
- Enable access to data from the network.
- Allows splitting archived variables into groups assigned to separate resources archived in separate files, in separate locations, separately configured and managed.
- Allows defining, from what station in the network the given group of data should be received.
- Allows automatic creation of backup copies of archive files, separately for each resource and each file type. Number of most recent files, that should be backed-up, can be set.

- By means of **AsixConnect5**, module it enables access to independent programs working under the Windows system.
- Allows preparing and later reconstruction of pattern trends.
- Allows operation in hot mode, completing the data from other stations with identical archive.
- Allows defining, what stations on the network should be granted access to gathered data.
- Gives access to software interface for dynamic configuration during **asix** system running.
- Allows setting parameters for trend archiving directly from the **asix** system variable database.

There is also a new program **AspadTools** in the package, which allows:

- changing the name convention from old (ASPAD 5) to new (ASPAD 6) and vice versa;
- conversion of the archive to different type (for example M to D);
- export to the text file;
- checking and repairing the archive;
- gathering statistics concerning the size of the data for particular archived variable trends;
- change of the name or deleting the variable from the archive file.

Description of the AspadTools program can be found in its help file AspadTools.pdf.

7.1.2. Basic Kinds and Types of ASPAD Archives

The basic type of the archive is **STANDARD** one. This archive is saved in binary disk files.

The second type of the archive is **ASBDE** and **MSSQL** one, this archive is saved in database – appropriately of Paradox or SQL type.

ASPAD stores the data in files containing trends from one day, month or year period, adequately to type of archive:
D-type archive (***d****ay*) – data collected in daily files;
M-type archive (***m****onth*) – data collected in monthly files;
Y-type archive (***y****ear*) – data collected in yearly files;
H-type archive (***h****orizon*) – data collected in one file;
B-type archive (*data**b**ase*) – data collected in typical databases ;
P-type archive (***p****attern*) – pattern trends.

ASPAD provides access to data from all periods, whose files are located in directories indicated during configuration.

The **H**-type archives (with *horizon*) are stored in one file for specified group of variables. Data older than the declared horizon are automatically forgotten, and the place in the file is recovered for further archiving. This type of the archive can be used for temporary archiving of additional trends, for example for a quick registration of some trends for diagnostic purposes. In current version it is recommended to use **D** archive, with files storing horizon set, for this purpose, and **H** archive has only been temporarily preserved to keep compatibility with previous versions of the ASPAD program.

The **B**-type archives collects values in typical databases. Currently these are databases in Paradox and SQL standard. It is suitable for recording values with low sampling rate giving high flexibility with using such database, and also a possibility of inserting historical data in the archive.

Except for listed ways of archiving also **pattern trends** can be generated and treated like archives. This type of trends is marked with **P** letter (***p**attern*).

You can also distinguish splitting the archive according to the way of data access:

WRITE_ARCHIVES	- basic archive allowing writing and reading the data from archive files;
READ_ARCHIVES	- archive allowing only reading the data from archive files;
NET_ARCHIVES	- archive allowing reading the data through the network from another computer;
SLAVE_ARCHIVES	- archive allowing reading and writing the data, used only as redundant basic archive on another computer.

Configuring of particular archives is described in: Architect user's manual, chapter *3.8.4. Historical data merging parameters.*

The ***SLAVE archive*** is an archive for write; it is placed on a supportive server of a given resource. This archive should ensure a redundancy of the basic archive placed on a basic server. A difference between the slave archive and the basic archive consists in preferences for the basic archive when connecting the network clients. The client will connect to the slave archive only if it does not find the basic archive.

The slave archive is used to provide temporary redundancy during the time of maintenance or failure on the basic server, e.g. it is possible to declare the basic archive for 100-day time period and the slave archive with 10-day horizon. It allows data replenishment of the basic archive from the period of a break (the break is not longer than 10 days), but the clients have to connect to the basic server, ensuring data reading within 100-day horizon.

7.1.3. Configuration parameters of Aspad module

The parameters of ASPAD are set up with use of Architect module.

See more in: Architect user's manual, chart *3.8. Configuration of archival data.*

7.2. ASPAD Modules

Among ASPAD modules are:

Aspad.dll	- basic module of the ASPAD program,
AsBDE.dll	- database access module – required version compatible with Aspad.dll module,
PEdit.exe	- pattern trend edition program,
BBrowse.exe	- database browser program,
AspadTools.exe	- archive files maintenance tools,
ASSQL.DLL	- module for access to database of SQL type.

7.3. System Requirements

ASPAD is designed for cooperation with other modules of **asix** system running under Windows XP/2000/NT/ME systems. It is delivered as DLL library file. Loading the library is performed automatically during **asix** loading.

To ensure correct operation of ASPAD it is necessary, to configure ASMEN program in order to deliver all archived data.

To ensure correct operation in the network, the ASLINK module provided with network software is also needed.

Additionally, working with B archive and P pattern trends requires also installing BDE (standard Borland software enabling access to databases) or MSDE (when SQL database is used). Listed modules are provided with the **asix** installation package.

BDE program configuration is described in *7.8.2. Additional Requirements for B-Type Archives*.

Methods of estimating the sizes of operating and disk memory, needed for operation of ASPAD archiving module, depending on the number of archived variables and archive types, are given in *7.3.1. RAM Memory.*

7.3.1. RAM Memory

The quantity of memory required for the ASPAD modules is a sum of code size and the data and allocated operating areas.

The area of dynamic allocated memory depends mainly on the number of archived variables. Each variable, stored in D, M or Y archive requires about 0.8 KB to 1.3 KB, but storing in B archive needs only about 0.3 KB. If the variable is stored in few ways, it must be taken into consideration separately for each archive.

The initial amount of used memory is usually 0.8 KB for D, M and Y archives for one variable, so it cannot be considered as the total demand for memory. In

case, when the buffer of 0.5 KB size allocated for each variable is full of data, the disk writing operation is initialized and the new buffer is allocated for new data. As a result, until the disk writing operation is completed, one variable takes 1.3 KB of memory. If the disk writing operation is initialized for more then one variable at the same time, the amount of used memory dramatically increases. The above effect takes place at the end/begin of each day, when the described operation is executed for all the samples of D type.

EXAMPLE

For data structures of 5000 variables in D archive and 5000 variables in M archive a total area of about 13 MB of non-paged RAM memory should be provided.

7.3.2. Disk Memory for the D, M, Y and H Archives

The amount of the required disk memory depends mainly on the number and size of stored records containing the consecutive items of all the archived samples of variables.

For the archive of H-type the room for data from the time period twice longer than the established time horizon must be ensured.

For the archive of D, M and Y type the reserved disk area must be large enough for data of such number of months that is going to be stored on the disk at the same time.

Moreover, the ASPAD stores information that is necessary for organization of the internal file structure. That additional information for the D, M, Y and H archives requires more then 10 per-cent of the archives file (about dozens KB for one variable in one file). The rate of additional information increases for the less number of data.

Evaluation of amount of disk memory required by data records cannot be precise, because those data are compressed, efficiency of the compression depends on the sampling type and on the parameters of the archive. Every record before compression consists of 1 status byte describing the record (for example distinguishing the hole from the data), 4 bytes of the time stamp, 2 bytes of variable's quality (OPC standard compliant) and 1 to 4 bytes of variable's value, depending on its type. Only those records that have significantly changed its value (i.e. more than one unit of the precision of registration established while setting of parameters) are stored to the file. For the slowly changing values (related to the established sampling rate) only a few records are to be stored. Most of time stamps are to be reduced to one byte or omitted at all during the compression, the marker is only left in the status byte. The compression is as better as lower precision of time registration is required. The compression enables also to reduce the most of values of the variable to 1 or 0 bytes. Only for values that are changed sharply and the change is much more than the established precision, the compression may be inefficient. The status byte can be also missed, whether the size of the reduced time marker and value of the variable are the same in the two consecutive records.

In the result, during the archiving of fixed-point or floating-point numbers with defined registration precision, typical size of the record after the compression is

1-2 bytes. The same size of the record is achieved for floating-point numbers without the registration precision set, with some of the ASMEN's conversion functions, for which the raw value is an integer number. For the other floating-point numbers typical record size is 3-5 bytes.

The gaps in archive are also compressed and are 1-2 bytes sized.

7.3.3. Disk Memory for the B Archive

In current design of the B-type archive the record requires 20 to 40 bytes depending on the choice of optional fields. Each record for the defined sampling rate is saved even if no significant change of the variable value occurred. In addition, the database records are indexed by value and time. Such indexes usually take more space than the database records.

For SQL databases is the similar situation. The database also contains, except data, indexes, fields limiting the records and null area resulting from the way of data spacing. Additionally, a log file is created. Estimated database size averages (without a log file):

- 142 bytes per sample for a whole record with all additional fields;
- 113 bytes per sample for a record without additional fields.

Therefore the B-type archive is not suitable for collecting a large amount of data. But the use of standard file format is its advantage. It is also the only archive, where missed data can be completed.

7.4. The Files Created by Programs of the ASPAD Module

The programs of the ASPAD module create themselves number of files of various purpose and place them in the directory defined in the configuration file. The files contain data, configuration information, archive files verification reports, copies of the original archive files after their possible repairing and diagnostic messages.

7.4.1. Archive Files of D, M, Y and H-Type

Splitting the Archive into Files According to the Time
Splitting Files into Resources
Filenames Convention
Examples

Splitting the Archive into Files According to the Time

Archives of D, M, Y and H-type are collected in files of four types, distinguished by time of data, collected in specific type of files. Every variable may be archived simultaneously in files of more than one type. For D, M and Y archives the files containing data from specific period are created every: 24 hours (D), month (M) or year (Y).

At the beginning of every period a new file with unique name is created. Files for previous periods are stored till the moment, when the time period, set in archive parameters, passes. Some of the files can be secured by software against automatic erasing, which is defined in **asix** system parameters. Such files can be deleted during system maintenance. The oldest files are also deleted automatically in case of disk space shortage. Previously, the warnings are sent periodically, which allows aware reaction, for example copying the archive to the backup disk or tape.

Such approach allows flexible splitting the archive into parts. The 24-hours files are the smallest, which makes their maintenance, storing or selective deleting easier. However usually using the 24-hours files makes sense only for often sampled and changing data. Using those files for example with hourly data is causes disk memory loss, because the variable declaration in a new file takes up to about 40 bytes, and the disk memory for the variable data is allocated in 512 bytes chunks. Saving 24 records, which take up 1-2 bytes per record after compression, would cause the necessity of taking up 552 bytes for saving a few dozens points.

An example of effective disk usage is archiving temporary data in 24-hours files, 5-minutes averages in month files, and hour averages in year files.

The H archive files are not divided into periods, and the place freed by the old data is used again by the new data.

Splitting Files into Resources

In the initialization file particular groups of variables are assigned to the named resources parameterized in **ARCHIVES**, and each resource is archived in separate files with localization declared separately for each resource. Name of the resource is a part of the filename.

Filenames Convention

Current filenames convention uses long filenames, allows encoding of any date and uses, accordingly to the Windows convention, uniform filename extension. It was introduced in version 6 of the ASPAD program, providing adjustment of the filenames to new ASPAD features and Windows system requirements.

The file name consists of a couple of elements, following this pattern:

*Tyyyymmdd***-***resource***.ahf**

where:

T	- one-letter name of the archive type (D,M,Y,H);
yyyy	- 4-digit year, for H archive replaced with **xxxx** text**;**
mm	- 2-digit month, for H and Y archives replaced with **xx** text;
dd	- 2-digit day of the month, for H, M and Y archive replaced with **xx** text;
resource	- name of the resource declared in the initialization file;
.ahf	**-** constant extension text (***A****SPAD* ***H****istorical* ***F****ile*).

EXAMPLES

M200005xx-BLOCK.ahf	means the month file from May 2000 for the BLOCK resource,
D20000416-DIV1.ahf	means 24-hours file from April 16th 2000 for the DIV1 resource.

7.4.2. Archive Files of B and P-Type

During the work of B and P archive, files made by the ASBDE.DLL module are created. They are characterized by .DB, .PX, .XG0 and .YG0 name extensions.

The basic data are stored in ***.DB files**. The rest are index files, which can be restored by the ASBDE module.

Due to the fact that those files are typical for PARADOX databases, they can be processed and browsed with any tools supporting this type of databases. However, to browse the database, it is better to use specially created the BBrowse program, which contains information about relations between particular tables.

When using the **SQL databases**, *ASSQL.dll* creates a database in the default directory of the SQL server. Database contents may be browsed by means of standard tools used for SQL databases.

7.5. Configuring the System During the ASPAD Module Installation

Before running the ASPAD module it is needed to:

- properly format the hard disk and install the Windows system– because of reliability and efficiency **it is recommended to archive on the NTFS partition only**,
- create ASPAD section in the application initialization file,
- create the rest of ASPAD's configuration files,
- properly parameterize the ASLINK and ASMEN programs, to enable cooperation with ASPAD, supplying it with data for archiving.

It is not necessary to manually create directories containing the archive, because ASPAD can create them by itself during startup.

7.6. Configuring and Starting the ASPAD Module

The ASPAD is loaded automatically while the asix system starts running and its work parameters are read from the ASPAD section of the application configuration file. Loading of the ASPAD can be omitted, if the *Simulation* parameter is set up with use of Architect module:

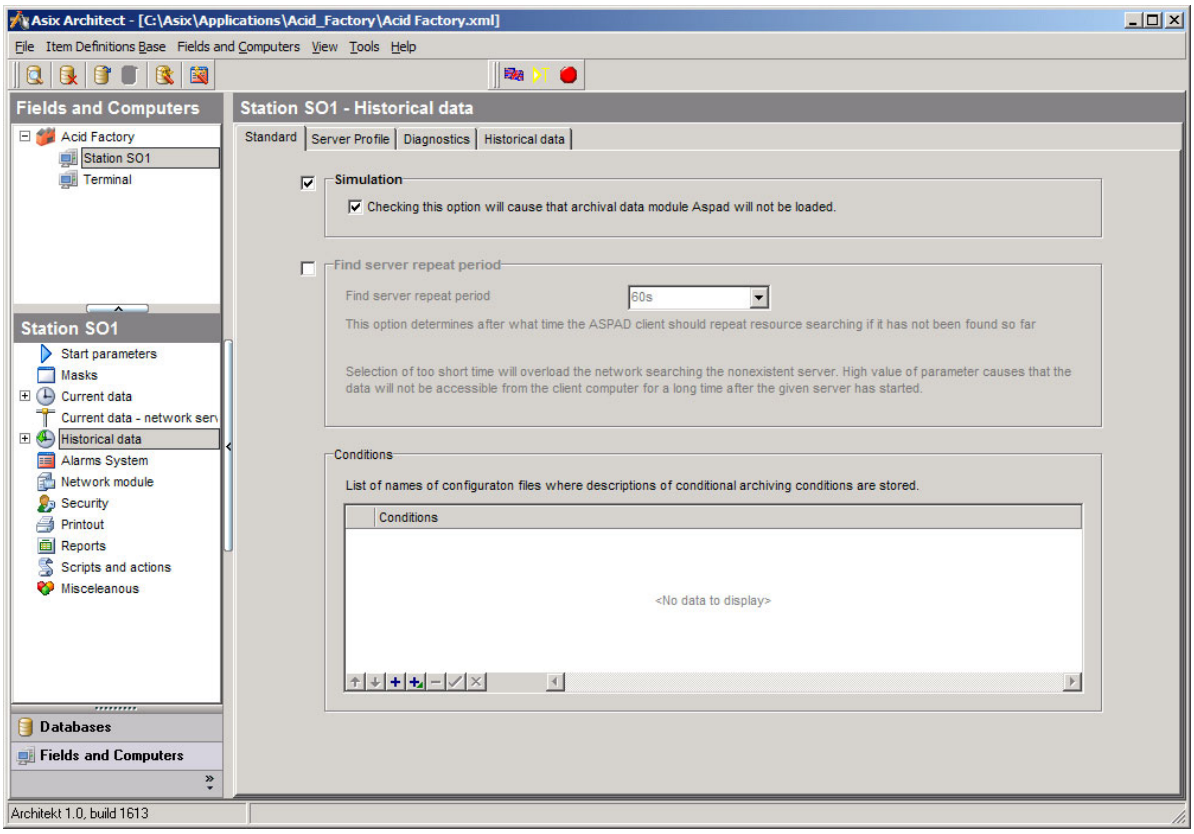


Figure 7-1. Configuration of Archival Data - Standard Data.

7.6.1. Specification of Time Periods in Configuration Files

There is often the need to define in configuration files either the time horizon, frequency or duration of executing defined actions. The uniform way of specification of the above parameters has been applied for setting parameters of the ASPAD. Time description is given in the following manner:

<number><unit> [<number><unit>[…]]

where:

<number>	- denotes the number of time units that are defined just after the number, the range of the number is from 1 to 32767,
<unit>	- defines the time unit, the following units are allowed: s - second m - minute h - hour d - day

EXAMPLE

3600s	or	1h			=	1 hour
1h30m	or	1h 30m	or	90m	=	1 hour 30 minutes
48h	or	2d			=	2 days

7.6.2. Definition of the Basic Parameters of Data Collecting

When the data collecting process starts there is the need to define some global parameters of it, which are set up with use of Architect module. The detailed descriptions of the individual declarations are given in: Architect user's manual, chapter *3.8. Configuration of archival data.*

The ASPAD's archive is split into resources defined in declarations:

Read archive,
Write archive,
Slave archive,
Network archive.

Run of variables for a given source are written and read into/from archive files placed in directories declared in above archives.

The archive is defined by the following parameters:

resource	- logical name of the archive,
type	- type of resource parameterized: STANDARD - archiving in files ASBDE - archiving in database MSSQL - archiving in database
additional parameters	- parameters dependant on the type of resource.

See more in: Archive user's manual, chapter *3.8.5. Archive parameterization*.

7.6.3. Automatic Backup

ASPAD can periodically create backup copies of the archive D, M and Y-type files on an additional disk. For this purpose the declaration BACKUP is used, which specifies, which files should be copied and when.

7.6.4. Automatic Deleting of the Oldest Files

It is possible to set in the ARCHIVE declaration the time, after which specified files of given resource should be automatically deleted. It gives the possibility of storing data, that should be deleted after specified time, expressed in days. It is recommended way of operation for storing data with the time horizon set. In previous versions of ASPAD program the H archive was used for that purpose. Current version accepts this type of archive only for keeping compatibility with previous versions.

Accidental, even momentary, serious change of computer time could lead to automatic erasing important files. To avoid such situation, after expiration of file validity, its name *.ahf is changed to *.ahr, and this file is still stored for another 10% of file validity period.

7.6.5. Access to ASPAD Data via Network

ASPAD, in the presence of ASLINK program, makes the data collected in its archive files accessible on the network (the condition is also having an **asix** key with server privileges). ASPAD acts then as the network server for those resources. Accessible data can be read from other computers in the network by means of ASPAD programs, for which the same resources has been specified in network archive parameters. Further they will be called network **clients of these resources**.

For the client, ***Find server repeat period*** defines how often to repeat the try of searching the server, if the resource currently is not available on the network. Parameter ***Time for Server Profile*** says how often, during the search for resources in the network, one should wait for server answer:

Architect program > *Field and Computers* > *Historical data* module > *Standard* tab:

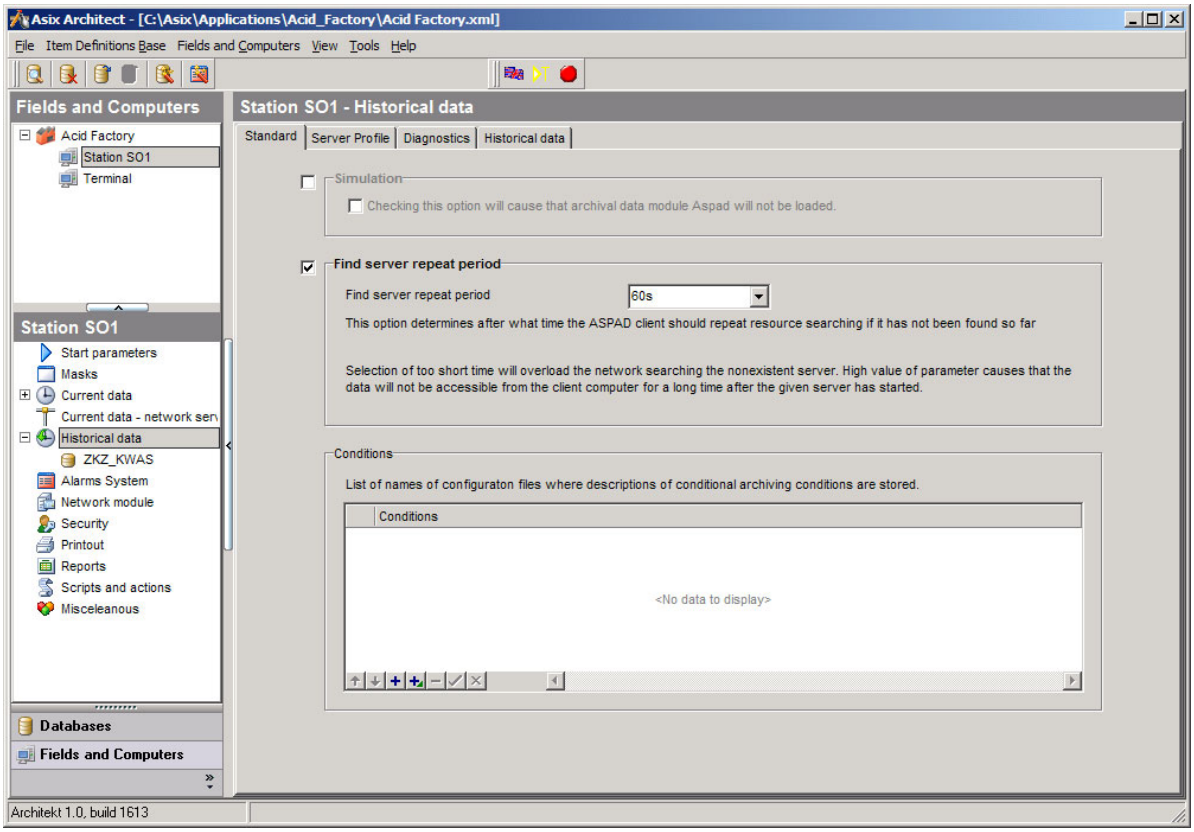


Figure 7-2. Configuration of Archival Data - 'Find Server Repeat Period' Parameter.

Architect program > *Field and Computers* > *Historical data* module > *Server Profile* tab:

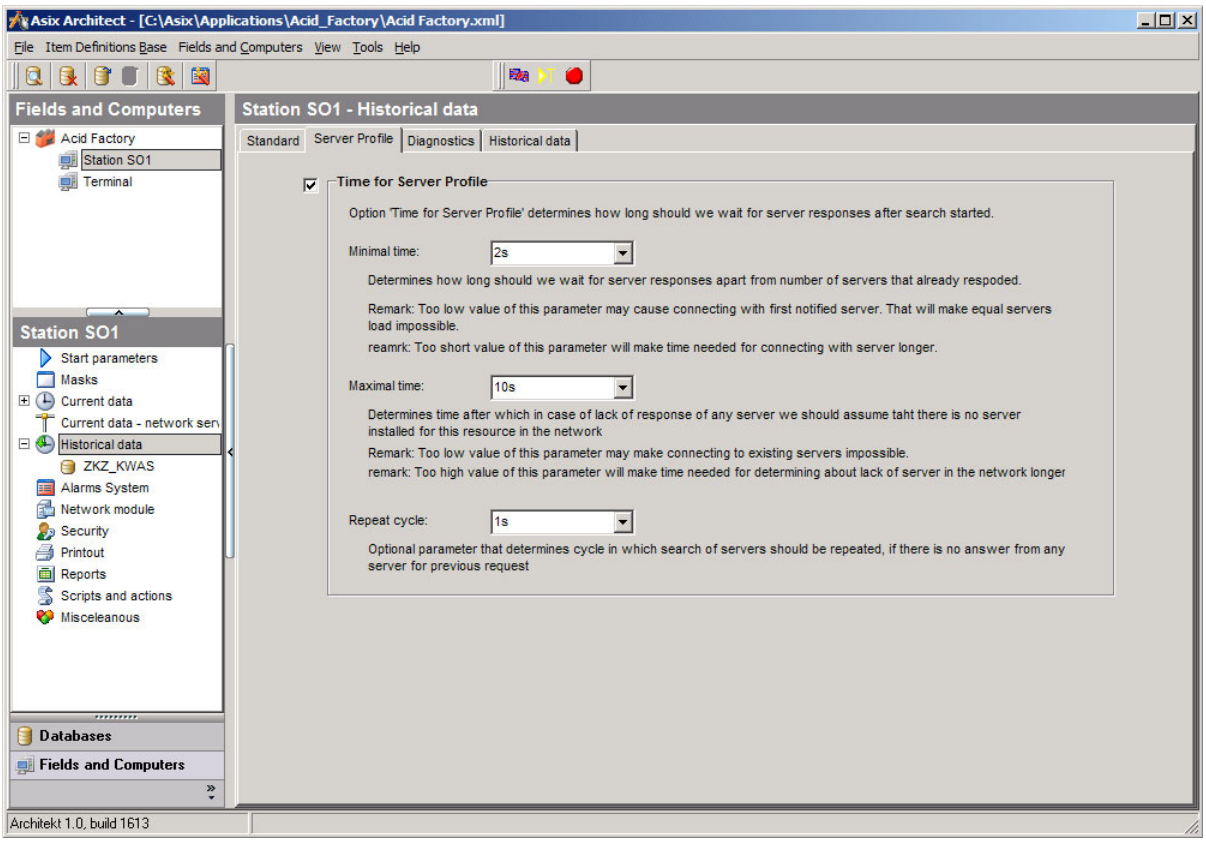


Figure 7-3. Configuration of Archival Data - 'Time for Server Profile' Parameter.

7.6.6. Variable Archiving Parameters

NOTICE *Archiving parameters are obtained from the variable definitions database (**VarDef**).*

Archiving parameters in variables database are contained in fields with following names:

Parameter	**Field**
Name	Name
Archive	Archive
Archiving parameters	ArchivingParameters

Described parameters characterize the way of archiving the variable and they are supplement of the rest of parameters set in variable definitions database.

Special attention should be paid to type of the variable, defined by the conversion function of the ASMEN module. Way of packaging the variable in the archive depends on its type. That's why **ASPAD can't archive data of another type than already collected in the archive, with the same variable name**. Let's suppose, that there was NOTHING (WORD type) conversion function declared for the variable. An archive of the WORD type variable has been created. Next, the conversion function was changed to NOTHING _INT (INTEGER type). ASPAD is not capable anymore of archiving this variable properly, and during startup produces

warnings about differences between the type of data declared in ASMEN module and the type of archived data.

Variable declarations has the following form:

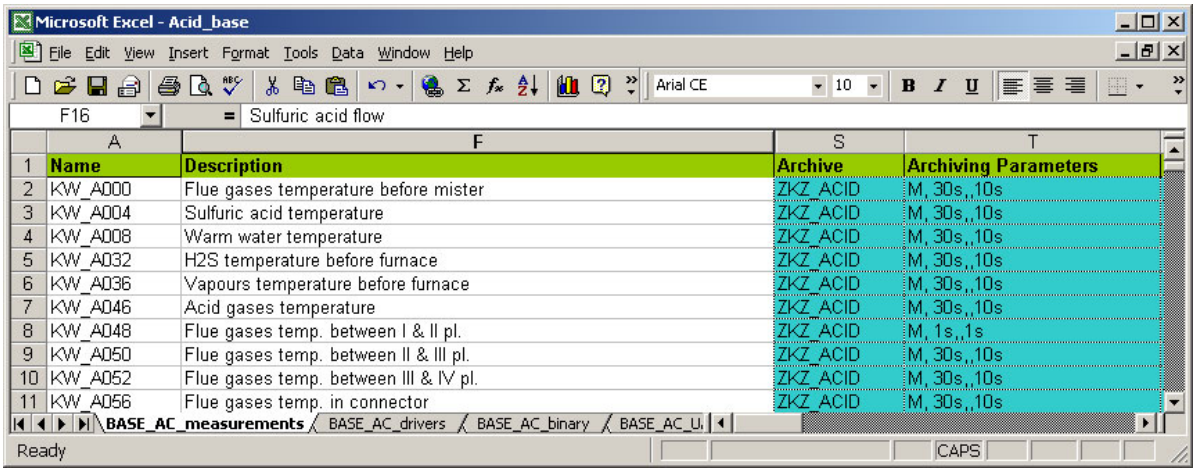
VName, ArcT, Period, Hor, Dt, Dx, Attr, Cond, BeforeCond, AfterCond

where:

VName	- name of the variable, compliant with the declaration in configuration of the ASMEN module;
ArcT	- one-letter definition of the archiving type;
Period	- variable's value sampling period; it should meet real needs, because too short period would cause increased disk space and bigger system's time consumption, and too long period could cause the value trend to be presented incompletely;
Hor	- minimal time horizon of storing data for H and B variable type; 0 value means infinite time, which is the default value; it is recommended to set finite horizon, because leaving 0 value can lead to exceeding the limit of 2GB for H and B archive files;
Dt	- time registration details level, for example 30s means, that either data received at 14:27:45 and at 14:28:14 will be registered with time 14:28:00; setting the biggest possible value for *dt* allows better packaging data in archive files; minimal *dt* is 1;
Dx	- data registration details level, which defines what changes of the variable value should cause saving the new value; with minor changes following changes will not be registered as unimportant, and during reading the previous value written in the archive will be valid in this place, for example 5 means, that new values should be registered only when the change regarding previously registered value is greater or equal to 5. 0 means registering with any change, omitting registering of the constant value; setting the biggest possible value for *dx* allows better compression of data in archive files;
Attr	- **Archiving attributes**; several attributes delimited with pipe ('|') can be specified;
Cond	- name of the optional **archiving condition**; definitions of archiving conditions are placed in files specified in CONDITIONS=... declaration;
BeforeCond	- the period before occurrence of *War* condition, the data from which should be saved to the archive; this parameter shouldn't be too big, because it requires creating appropriate buffers in RAM, including the number of data samples for the specified period;
AfterCond	- period after expiration of the *Cond* condition, the data from which should be saved to the archive.

Individual fields of declaration are separated with commas. Two commas, not separated with the parameter's value mean accepting the default value. Omitting a few of the last parameters will assign them default values.

EXAMPLE



	A	F	S	T
1	**Name**	**Description**	**Archive**	**Archiving Parameters**
2	KW_A000	Flue gases temperature before mister	ZKZ_ACID	M, 30s,,10s
3	KW_A004	Sulfuric acid temperature	ZKZ_ACID	M, 30s,,10s
4	KW_A008	Warm water temperature	ZKZ_ACID	M, 30s,,10s
5	KW_A032	H2S temperature before furnace	ZKZ_ACID	M, 30s,,10s
6	KW_A036	Vapours temperature before furnace	ZKZ_ACID	M, 30s,,10s
7	KW_A046	Acid gases temperature	ZKZ_ACID	M, 30s,,10s
8	KW_A048	Flue gases temp. between I & II pl.	ZKZ_ACID	M, 1s,,1s
9	KW_A050	Flue gases temp. between II & III pl.	ZKZ_ACID	M, 30s,,10s
10	KW_A052	Flue gases temp. between III & IV pl.	ZKZ_ACID	M, 30s,,10s
11	KW_A056	Flue gases temp. in connector	ZKZ_ACID	M, 30s,,10s

Figure 7-4. An Exemplary Declaration of Variables in VarDef.

Is equivalent to below declarations.

```
# Save the KW_A000 variable in M-type archive,
# M-type archive (1-month files),
# receive data every 30 second,
# register the time with an accuracy of 10 second,
# register every change of variable value,
# don't initiate the ASMEN variable value.
KW_A000, M, 30s, , 10s,

……

# Save the KW_A048 variable in M-type archive,
# M-type archive (1-month files),
# receive data every 1 second,
# register the time with an accuracy of 1 second,
# register every change of variable value,
# don't initiate the ASMEN variable value.
KW_A048, M, 1s, , 1s,
```

7.6.7. Atributes of Archiving

RESTORE
During startup ASPAD writes to the ASMEN module database the last value registered in the archive. It allows storing and retrieving of the values set by **asix** system operator also when the computer is turned off. Attribute is generally intended for variables belonging to NONE type ASMEN's transmission channel (no variable refreshing by reading from the controller). From the moment of ASPAD module start the values set before turning off will be accessible for all ASMEN module users in given system.

DONT_RESTORE cancels the RESTORE declaration.

NO_PACK
NO_PACK attribute changes two archiving features.

In the archive, in every archiving cycle, writing the value will be done, on the condition, the correct value will be obtained. So writing of the constant value will not be omitted in the following cycles.

During the read ASPAD will be set up for reading the points, not the segments. In normal mode ASPAD makes segments accessible, which means, that if value was written with time 1:00 and is valid for one hour (sampling cycle is 1h), then during startup at 2:00 a new value will appear, and if it doesn't exists (a gap in the archive), a value from 1:00 will be repeated there. It allows drawing a segment from 1:00 h till 2:00 h and showing the hole in the archive yet from 2:00. NO_PACK attribute will cause reading only the value from 1:00.

PACK cancels NO_PACK declaration.

ALL
ALL attribute can be used only with B-type archive. Additionally, the variable must be obtained from the channel, whose driver provides additional services allowing to receive every incoming value. Currently the SINAUT driver has such capabilities.

ALL attribute will cause writing to the B-type archive all values received by driver, apart from the sampling cycle and the order of giving access to them. The sampling cycle then only means the time of validity of received data.

NOT_ ALL cancels ALL declaration.

7.7. Conditional Archiving

Conditional archiving allows archiving only when specified conditions are met. It can be set, data from what period before occurrence of the condition and after its expiration should be saved in the archive.

The declarations of conditions occur in files given in CONDITIONS=… declarations.

7.7.1. Declaration of Conditions

Archiving conditions are defined with expressions in files specified in CONDITIONS line. Every expression must be defined in a separate line.

Values and arguments of expressions are constant or variable floating-point values, 32-bit masks, or logical values. Expression arguments can be other expressions, which means that complex expressions can be defined. Names of expressions can't be repeated. If the expression value is not a logical value, and is used directly as a condition, a conversion is performed:

- 32-bit masks, where all bits are zeroed, are converted to false, other to truth,
- numbers for range(-0.5;0.5) are converted to false, other to truth.

Expression line has following structure:

Expression_name, Operator, Arg1, Arg2, ...

where:

Expression_name	- expression identifier containing up to 15 characters; name of the expression cannot be repeated; expression names are a separate group, and thanks to it conflicts for example with identical variable name can be avoided;
Operator	- name of one of the acceptable operators;
Arg1, Arg2, ...	- expression arguments.

Arguments have following form:

[*u*][**$**]*source*

where:

u and **$** - are optional and mean:

u - is one of one-argument operators:
– means change of the sign with previous conversion to the floating-point number,
~ means negation of mask bits with previous conversion to the 32-bit mask,
! means negation with previous conversion to the logical value;

$ - means taking the value calculated during previous valuating of the condition for the specific variable and keeping current value for future calculations.

The *source* of data has one of the following forms:

decimal_number	- numeric floating-point value (e.g. 9999);
0x*hexadecimal_number*	- 32-bit mask encoded hexadecimally (e.g. 0xFFFF);
>>	**- p**lus infinity;
<<	**- m**inus infinity;
@*variable_name*	- value of the process variable;
@	**- v**alue of the process variable, the condition is currently being specified for. Such condition has different values depending on the variable it is specified for;
?*expression_name*	- *v*alue of another, previously defined expression. Complex expressions can be defined in this way.

7.7.2. Archiving Condition Operators

Logical Operators
Relational Operators
Logical Bitwise Operators
Relational Bitwise Operators
Arithmetic Operators
Range Test Operator

Condition operators can have various types of results and various number and type of arguments, depending on the type of operator. In case of incompatibility of argument type with the type accepted by the operator, a conversion is made automatically. Below currently defined operators has been described.

Logical Operators

OR	logical sum
AND	logical product
XOR	exclusive logical sum
Arguments type:	logical value
Result type:	logical value
Number of arguments:	any

Relational Operators

LT	less
LE	less or equal
EQ	equal
NE	different
GE	greater or equal
GT	greater
Arguments type:	numeric value
Result type:	logical value
Number of arguments:	2

Logical Bitwise Operators

BAND	logical product of all bits of arguments
BOR	logical sum of all bits of arguments
BXOR	logical exclusive sum of all bits of arguments

Arguments type:	32-bit mask
Result type:	32-bit mask
Number of arguments:	any

Relational Bitwise Operators

BLT	sets bits given in *Arg3*, which in *Arg1* are < than bits in *Arg2*
BLE	sets bits given in *Arg3*, which in *Arg1* are < or = to bits in *Arg2*
BEQ	sets bits given in *Arg3*, which in *Arg1* are = to bits in *Arg2*
BNE	sets bits given in *Arg3*, which in *Arg1* are not = to bits in *Arg2*
BGE	sets bits given in *Arg3*, which in *Arg1* are > or = to bits in *Arg2*
BGT	sets bits given in *Arg3*, which in *Arg1* are > than bits in *Arg2*
Arguments type:	32-bit mask
Result type:	32-bit mask
Number of arguments:	2 or 3. If only 2 arguments are given, it is assumed, that *Arg3*=0xFFFFFFFF.

Arithmetic Operators

ADD	arithmetic sum
MUL	arithmetic product
DIV	arithmetic quotient *Arg1* / *Arg2*.
Arguments type:	numeric value
Result type:	numeric value
Number of arguments:	ADD and MUL - any, DIV - 2.

Range Test Operator

IN	fulfilled, when value of Arg1 is within the range [*Arg2*, *Arg3*], or in the range [*Arg4*, *Arg5*] etc.
Arguments type:	numeric value
Result type:	logical value
Number of arguments:	odd number not less than 3.

7.7.3. Condition Examples

Archive, when 4. or 8. bits of AAA variable has changed their status:

```
CONDITION, BNE, @AAA, $@AAA, 0x1100
```

Archive, when bit 0. of archived variable changes its status from 0 to 1:

```
CONDITION, BGT, @, $@, 0x1
```

Archive, when value of the archived variable is out of range (20,80) and one of the bits 0-3 of WWW variable is set:

```
W1, IN, @, <<, 20, 80, >>
W2, BAND, @WWW, 0xF
CONDITION, AND, ?W1, ?W2
```

Archive in the moment, when the value of archived variable exceeds the limit of 100:

```
W3, GT, @, 100
CONDITION, AND, ?W3, !$?W3
```

7.8. B-Type Data Collection

7.8.1. Special Features of B-Type Data Collection

The B-type archive enables to make use of typical **databases** for storage of the ASPAD archive. On the one hand it makes possible to take advantages of that archive in more flexible manner and preserves opportunity to use it also by programs that are not included in the **asix** package. It is the only collecting method that enables **filling gaps** that have appeared in the archive. On the other hand no form of data compression or omitting unimportant changes of values is possible and each record takes over **50 bytes** (having considered index files).

asix4 enables archive storage in databases of Paradox and SQL environment.

Because reading and writing to B-type archive is slower, and files are of bigger size than in D or M archives, sensible solution is to use for example D-type archive for registering data for the charts, and B-type archive for storing aggregated data (e.g. hour averages for reports).

B-type archive also enables:
- archiving of additional information about archived points,
- archiving of particular variable groups in multiple databases,
- insequential archiving (values can come in any order),
- automatic archive indexing and database name in Paradox and SQL environment,
- browsing the contents of B-type archive databases with BBrowse browser and other standard tools used to browsing of Paradox and SQL databases.

7.8.2. Additional Requirements for B-Type Archives

Data Archiving with PARADOX Database
Data Archiving with SQL Database

Data Archiving with PARADOX Database

To enable B-type archiving, additional library ASBDE.DLL should be loaded by declaration of AS BDE archive:

Architect program > *Fields and Computers* > *Historical data* > add new definition with the following parameters:
- archive name;
- kind of Archive: AS BDE,
- additional parameters.

It is also necessary to install BDE software.

Next step should be configuring DBE for B-type archiving purposes. It can be done with help of BDE Administrator:

in Control Panel – BDE Administrator – Configuration – Drivers – Native – PARADOX
must be set:
BLOCK SIZE to 32768
LEVEL to 7

Data Archiving with SQL Database

System software:
Connecting with databases demands:

- MDAC in 2.7 version (or newer) for database client (a station with the **asix** system);
- Microsoft SQL Server 2000 for database server;
- database user have to possesses an account, on the SQL server station, ensuring writing to database;
- a disc with a database can not have an active compression.

asix software:
Access to database is realized by ASPSD.DLL (from 6.25 version) with ASSQL.DLL (from 6.00 version).

To enable B-type archiving, it is necessary to load an additional ASSQL.DLL library by declaration of MS SQL archive:

Architect program > *Fields and Computers* > *Historical data* > add new definition with the following parameters:
- archive name;
- kind of Archive: MS SQL,
- additional parameters.

User authorizations:
User authorizations are verified on Windows level. So authorizations of database access on SQL server computer should be assigned to the current user.

During first **asix** startup (when the proper declaration has been placed in the initialization file) the database is created – but only when the user has authorizations for database creation.

These notes refer to all databases declared for ASPAD in application configuration file, also to ones used only in SYNCHRO declaration (*See: 3.8.5.4. Declaration of synchronization of B-type archive contents with MS SQL base data*).

Declaration of share on the server:
Writes to the archive as well as synchronization need the temporary direction to be created on the server computer (to which the write is performed) as the share named ASSQL. It is necessary to make an access to temporary direction, where the writes will be performed, for all users. This direction is used to exchange of files that allow multiple increase of write rate to database.

It is also necessary to create ASSQL share when database is placed on the same computer from which the write operation is performed.

NOTICE *BDE and SQL database must not be simultaneously declared for the same resource!*

7.8.3. Replenishment of Data in the B-Type Archives

The B-type archives allow replenishing data in gaps that have appeared as result of missed links to a data source or turning the computer off. For making possible execution of such replenishment of data it is necessary, that the device, that is a data source, would make available not only the current data, but historical data too. The ASMEN together with drivers of that device must make available access to these data.

The parameters ***Time of first merging*** and ***Merging frequency*** let to define, when such replenishment should be executed:

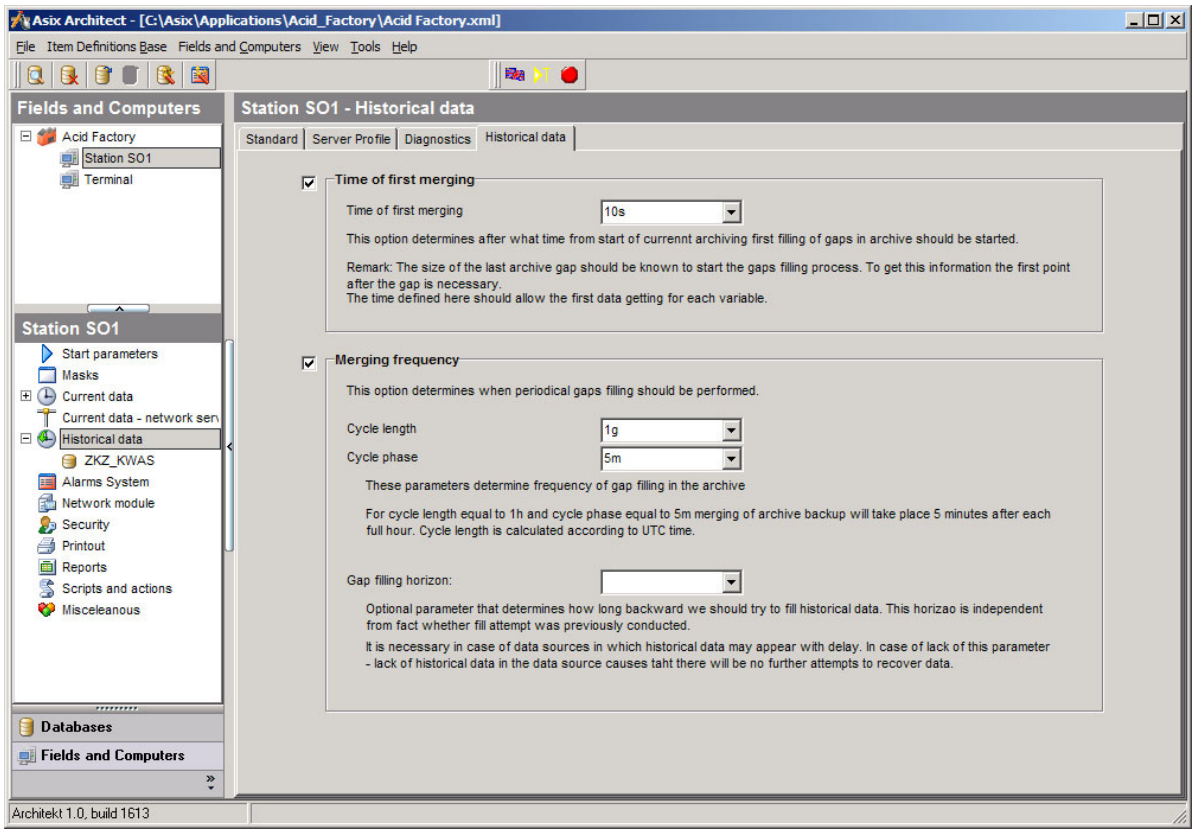


Figure 7-5. Parameterization of Replenishment for B-Type Archive.

7.8.4. Synchronization with a Dual Database

Advices
Creation of Follow-Up Archive on Mediating Server
Mutual Replenishment of Operator Station Data
Making the Copies of Archive

Function of synchronizing the archive with dual database allows on-line data replenishment, that ensures replenishment of historical data and manual data edition.

The synchronization causes incremental copying. Only the data that have been escribed or changed in source archive from the last copying. In case of conflict with existing data, the data with better quality are stored in the destination file. In case of data with the same quality, manual entry data are preferred.

Many SYNCHRONIZE declarations may be set, also for one source. It allows e.g. replenishing data from several dual servers.

Details referring to SYNCHRONIZE, SYNCHRO are in:
Architect user's manual, chapter *3.8.5.4. Declaration of synchronization of B-type archive contents with MS SQL base data*.

Advices on Using SYNCHRO (SYNCHRONIZE) Declaration

SYNCHRO declaration can be used for many reasons:
- mutual replenishment of operator station data;
- creating a follow-up archive on mediating server;
- making the archive copies.

Mutual Replenishment of Operator Station Data

SYNCHRO declaration allows mutual replenishing data between operator station. It has a few advantages in comparison with a typical solutions in ASPAD module:
- it allows on-line data replenishment, not only during startup;
- it replenishes all changes in archive, including on-line replenishment of both historical and manually written data.

Creation of Follow-Up Archive on Mediating Server

The source only for read may be declared on mediating server, and then its synchronization from operator stations can by set. Such a server will not take the current data form the ASMEN module. The data will be received from operator stations with database synchronization in cycles. The synchronization will includes all escribed data, also historical.

Making the Copies of Archive

SYNCHRO declaration can be used for making the reserve archive. The copy should be placed on other than main station. The copy will be fully functional database.

Any **asix** need not operate on SQL server where the copy is created. The only purpose of SYNCHRO operation is to receive a reserve database.

Such a kind of copy can be used only on a limited scale, because in comparison with reserve copy created by SQL system, it has a few faults:

- it has a large size, because contains transaction log files, index files and null area that arises from the way of data arrangement on SQL server;
- it may not be created on the disc with compression.

7.8.5. Database Declaration

It is necessary to declare the proper archive source, archive direction (in case of PARADOX database), database and server name (in case of SQL database) in the database (of PARADOX or SQL type) when archiving is applied.

To use a database in the **asix** system, it is needed to assign the database access authorizations to the current user (the proper authorizations on server are needed).

During the first **asix** system startup the database is automatically created and the user need to have proper authorizations for database creation.

Details concerning the database declaration are in:
Architect user's manual, chapter *3.8.5. Archive parameterization*.

7.8.6. Declaring the Variables for Archiving in Database

The declaration of variables for archiving in database is made by means of the proper parameters of variable archiving (*See: 7.6.6. Variable Archiving Parameters*).

Such a declaration is similar to archiving of variables both in PARADOX and SQL databases.

7.8.7. Additional Information about Archived Points

Typically, value and time of the point are archived, and the archive can be parameterized in the manner allowing enhancing the record with additional

fields. B-type archive allows saving the following set of fields (labels used in ARCHIVE declaration are given in the brackets):

value (X)

Number specifying variable value.

basic time (T)

In archiving with the sampling cycle set it's the time of beginning the following cycle. It is defined in following way:

time of the data is rounded to the closest multiple of registration details level set, for example with registration details level set to 5 minutes 1:17:30 will be rounded to 1:20:00, 1:22:00 to 1:20:00, and 1:22:30 to 1:25:00;

time rounded in this way is rounded down again to the multiple of sampling cycle, for example with 15-minutes cycle time 1:20:00 and 1:25:00 will be rounded to 1:15:00.

In archives without the cycle assigned (**ALL** attribute) it is the time of the data rounded only to the set registration details level.

time of the data (XT) – optional field

Exact time assigned to the variable, rounded with accuracy set during parameterization. If the archive was given the **ALL** attribute, then it is equal to the basic time.

write time (WT) – optional field

The time of data write to database. This filed is important in systems with delayed data obtaining.

status (S) – optional field (recommended)

Status containing following information:

bits 0-7 – status compliant with OPC standard,

bits 8-9 – data source:

0 – data received in normal mode,

1 – data received as historical,

2 – data entered manually,

3 – simulated data,

bit 10 – data source location:

0 – local computer,

1 – copy of the data from another computer.

Data Archiving in SQL Database

SQL database of B-type archive may contain the following fields:

1. Obligatory fields:
   - basic time (**T**),
   - datum quality;
   - value (**X**).
2. Optional fields:
   - time of the data (**XT**),
   - write time (**WT**),
   - validity period of sample (**VT**).

validity period of sample (**VT**) - determines whether validity time of period will be archived in database (+VT), or it will be omitted and the default validity time will be the archiving period (-VT); if the

archiving period will be shortened in the future for –VT, then historical data will be separated by gaps; by default the validity time is archived.

Switching on the option causes adding 4-byte field to each data record.

Details concerning the parameterization of additional information of archived points are in:
Architect user's manual, chapter *3.8.5. Archive parameterization*.

7.8.8. Archiving in Multiple Databases

Multiple repeating the ARCHIVE declaration for various databases and assigning every group of variables to the specific database is possible.

NOTICE *BDE and SQL database must not be simultaneously declared for the same source!*

7.8.9. Insequential Archiving

After setting the ALL attribute, all the data received by ASMEN's driver will be saved to the B-type archive, apart from the order of receiving them.

7.8.10. Automatic Creation of the Archive Directory and Database Name in Paradox Environment

It is not needed to create directories of database archive. ASPAD will create them by itself during first startup.

ASPAD will also create a Paradox database name set in parameters. Name will be accessible from the Paradox database tools and from the dedicated **BBrowse** browser.

7.8.11. Browsing the Contents of B-type Archive Database with the BBrowse Browser

There is **BBrowse** browser included in the package, which enables browsing the B-type archives database contents. The access to the database is gained after choosing database name from the list of all B-type archive names on the given computer. User interface allows navigating in archive tables in an intuitive way.

Database contents may be browsed by means of standard tools used for SQL databases.

7.9. Pattern Trends

The pattern trends are specific type of the archive, in which the data is collected only in the moment of application parameterization. There is a pattern written in it, which can be later retrieved, with any time shift. They are treated as an additional, read-only archive type, marked with letter P.

Pattern data is stored in database analogous to B-type archive. The best solution is to place the patterns in a separate database, treated as a separate ASPAD program resource. It is possible to place B and P archives in the same database, but it's not recommended.

7.9.1. Pattern Trends Declaration

The pattern trend is marked as **P-**type. Declaration of such trend cannot be placed in writable archive – it has to be declared as read-only archive or network archive.

For the read-only archive, containing P-type trends, a declaration of database is necessary (Read archive declared as AS BDE or MS SQL), just like for the B-type trends.

Declaration of the pattern trend has the following form:

*trend_name***, P**

where:
*trend_*name shouldn't be the name of any of **asix** variables.

The data validity time, for the P-type trend, is always set to 24-hours period, and the other parameters are unimportant.

7.9.2. Generation of Pattern Trends

For generating pattern trends the **PEdit** program is used.

Detailed description of PEdit program usage can be found in file PatternTrends.pdf.

8. ASLINK - Network Module

The ASLINK is the module that enables communication between various components of the **asix** system by means of the local network (LAN). The ASLINK defines a set of network services used by other programs of the **asix** system, which are configured to use the network.

The ASLINK uses services provided by the NETBIOS emulator, i.e. the program that implements one of the most common and widespread used network standards on the transport layer. Specification of the NETBIOS was developed by the IBM Company, the ASLINK is based on the specification of the NETBIOS v. 3.0 included in the IBM's Local Area Technical Reference (IBM Part Number SC30-3587-00). The backgrounds of the operation of the ASLINK are NETBIOS services of so called link-oriented (session-oriented) data transmission.

The version of the ASLINK designed for Windows uses the standard NETBIOS interface provided by the operating system.

Installation of network services is not necessary for such configurations of the **asix** system, where access to the computer network is not required.

8.1. General Characteristics of the ASLINK Module

8.1.1. Services of Reliable Data Transmission

The basic service provided for such components of the **asix** package as, among others, ASMEN and ASPAD and sub-system of alarm handling, is reliable data transmission between every pair of processes operating on any stations connected to the computer network and handling the common user's protocol. For this area, the services of the ASLINK module aim to support communication based on existence of the *"server"* process as well as the *"client"* one that uses server services. In details, the ASLINK provides:

- searching for the processes in the network that can perform server function related to the resources needed for the client processes,
- establishing the link (so called channel) between the client and the server of the resource pointed by the first one,
- reliable duplex data transmission in every channel,
- information for the processes that communication channel has been closed because of the network failure or for direct order of one of partners.

8.1.2. Time Synchronization Service

In case of the **asix** network installation the time synchronization of every station over the network is very important. For this purpose at least one workstation should be provided with high quality source of current time and act as a timeserver for other network workstations. This is why the function of time synchronization has been implemented within the ASLINK network module.

Depending on parameters having been set for the ASLINK, it can perform function of a timeserver or it can synchronize only its own time with active timeserver. A priority is assigned to every time. At the moment only the timeserver of the highest priority is active. Logging to the network a new server of higher priority causes the current timeserver to change its mode to the ordinary synchronization mode, i.e. it will receive time from the new server. Turning the current server off (or its failure) causes activation of one of the other servers. The other parameter that characterizes operating of the new server is frequency that the server broadcasts its time in all over the network. All the workstations, that receive the current time broadcaster by the active server, execute synchronization of its time if the discrepancy between their own time and the server broadcasting exceed the maximum value defined by the appropriate parameter.

8.1.3. Remote File Access Services

The ASLINK module provides services related to file transfer between network workstations. Based on these services remote file browsing, copying a file onto a local disk, file writing onto the remote disk and file deleting are possible in the network of the **asix** stations. The options related to file writing/deleting might be granted separately for individual stations. For this purpose the parameters of the ASLINK module can be set in the appropriate manner.

8.1.4. Multi-Board (PCB) Configurations

The ASLINK is able to operate with a number of network boards simultaneously. It does not enable direct link of computers of two different networks. Data transmission between networks needs additionally to run on the gateway computer specifically configured ASMEN and ASPAD.

8.2. Hardware and Software Requirements

8.2.1. Types of Supported Local Area Networks (LANs)

The ASLINK can operate in most well known types of networks such as Ethernet, Token Ring, and ARCnet. The only condition of running the ASLINK program within the defined computer network is availability of the NETBIOS emulator that operates with software driver of the network boards used for the given network. This requirements is met automatically when the up-to-date network boards are used, because their software drivers usually match the NDIS specification proposed by Microsoft and accepted as the common method of board driving for many NETBIOS emulators.

In case of Windows operating systems the ASLINK can run within any network accepted by the system. The only condition is installation of the NETBIOS interface being the internal part of those systems.

Connection of a station by means of the RAS service (access through a modem) is also possible.

8.2.2. Software Requirements

Both network software and the NETBIOS interface have to be previously installed in Windows operating systems. The above components are internal parts of those systems. Network software developed by other companies can also be installed.

The only condition of correct operation is that that software should grant the NetBIOS interface.

Please remember that the following conditions have to be met for each case.

- For networks based on the Windows operating systems the configuration of these systems is necessary in such a manner, that the NETBIOS interface that runs on each station should be referred to the same protocol. As such a protocol the "NetBEUI protocol" or "NWLink IPX/SPX Compatible Transport" for Windows NT can be used.
- Setting parameters for the NETBIOS protocol is not required for Windows operating systems. The attention should be only paid to matching the ASLINK program to the appropriate NETBIOS interface, i.e. parameter ADAPTER within the initialization file should be properly selected. That parameter defines logical number of the network interface. For Windows operating systems, that include one (or more) network board, there can exist few logical network interfaces (logical adapters). Selection of the appropriate logical interface decides about the protocol that is destined to be the basis for communication engaging NETBIOS interface. The Windows NT enables to assign NetBIOS interfaces to the protocols in any manner. For Windows 2000, setting a protocol as the default one causes, that the related NetBIOS interface is assigned as logical number 0.

8.2.3. Remote Access Service (RAS) Activation in Windows XP/2003

The EnableRASLana program enables the Aslink network module to establish a connection in Windows XP and Windows 2003 using Remote Access Service (RAS), TCPIP protocol and NETBIOS interface. RAS allows to connect with remote station through modem, serial link, for example, etc. Some of that kind connections are blocked in the mentioned operating systems by default – it makes using them by Aslink impossible. To enable it, the EnableRASLana program should be run one time. Another running the program can be necessary after network configuration change.

Running the EnableRASLana program demands administrative authorities.

8.3. Configuration of Aslink Module

Parameterization of Aslinik module is realized with use of Architect module.

See: Architect user's manual, *chart 3.10. Configuration of network module.*

8.3.1. Station Identification

Stations are identified by names. Every station should be given a unique name in the network. Except for the individual name, every station belongs to the group of stations, having common group name. Between stations belonging to different groups data exchange is not possible. The same goes for time synchronization – the time of one station is synchronized inside of one group only. Exception to the above rule are stations working under control of the networking module version lower than 5.00.000. Network module version 5.00.000 or higher, configured to work with stations with lower versions will be able to make connections and synchronize time with them. Stations working under control of older versions of the network module belong to all groups.

The *Name* parameter is used to give the station its unique network name. The *Group* parameter assigns the station its group name.

8.3.2. Access to the Network

The ASLINK network module enables access to network via logical adapters. Logical adapter is a physical network adapter and a set of network protocols. Logical adapters are identified by their numbers or names. In Windows NT system, the logical number binding with protocols and physical network adapters can be determined with network settings of the operating system. These bindings can be also read with AslView program. In Windows 2000 and XP system the information about bindings can be only obtained with AslView program.

The set of logical adapters is given with the ***Network protocols*** parameter. The values of this parameter are logical adapter numbers or names separated with commas. If the „*" character is given as the value, all the logical LAN adapters, available in the operating system, will be used (for WAN – Internet and RAS – adapters specify "*,WAN"). General form of the *Adapters* parameter is described at the end of this chapter.

In some applications it is necessary to reduce the set of stations, which can be interconnected. This is performed with use of ***Logical adapters used by network module to search for servers*** parameter. The adapters, on which the servers will be searched, are given as the value. The parameter affects only clients on local station. It doesn't limit connections, initiated by remote clients. Lack of this parameter or passing the '*' character as its value will

result in using all adapters to do the search.

Another way of limiting the set of stations, that can be connected, is using ***Station name filtering*** parameter. The value of this parameter is a comma-separated list of names of stations, which can be connected to. The parameter affects only for making connections to remote servers. Clients on remote stations can connect to servers on local station without restrictions. If the filter name contains '?' character, remote station name will not be compared with filter in this characters' position. For example, if there are stations named "A1X", "A2X", and "B1X" on the network, and the filter name is "A?X", connections to servers on stations "A1X" and "A2X" will be possible, but not to "B1X" station. The filter name may contain the '*' character. In such a case, names of the filter and station are compared only in the part of the name preceding the '*' character.

EXAMPLES

The adapters defined in the system:

Table 8-1. Logical Adapters of Aslink.

Logical Number of the Adapter	„Network path" Defined in the System
0	NetBT->Tcpip->Realtek RTL8029(AS) PCI Ethernet Adapter Board
1	NetBT->Tcpip->Realtek RTL8029(AS) PCI Ethernet Adapter Board #2
2	NetBT->Tcpip-> 3Com EtherLink III ISA (3C509/3C509b) Board in Legacy" mode
3	Nbf->Realtek RTL8029(AS) PCI Ethernet Adapter Board
4	Nbf->Realtek RTL8029(AS) PCI Ethernet Adapter Board #2
5	Nbf->Tcpip-> 3Com EtherLink III ISA (3C509/3C509b) Board in Legacy" mode
6	Nbf->WAN Miniport (NetBEUI, Dial Out)
7	Nbf->WAN Miniport (NetBEUI, Dial Out)
8	Nbf->WAN Miniport (NetBEUI, Dial In)
9	NwlnkNb

Selected protocols = NBF
all logical adapters with Nbf protocol, but without WAN – type links

Selected protocols = Realtek&#2
all logical adapters bound with "Realtek RTL8029(AS) PCI Ethernet Adapter Board #2"

Selected protocols = Nbf&Realtek, IPX
all logical adapters bound with Nbf protocol and Realtek boards and all logical adapters bound with IPX protocol (NwlnkNb)

Selected protocols = tcpip
all logical adapters bound with TCP/IP protocol, but without WAN – type links

Selected protocols = tcpip&3Com
logical adapter 2 (TCP/IP protocol on 3Com board)

Selected protocols = NETBEUI&In
logical adapter 8 (incoming WAN connection)

Selected protocols = - TCPIP&3Com
all adapters except for logical adapter 2 (TCP/IP protocol and 3Com board) and WAN – type links

Selected protocols = - IPX
all logical adapters except for adapters bound with IPX protocol and WAN – type links

Selected protocols = - /2
all logical adapters except for adapters bound with board No. 2 (slot number) and WAN – type links

Selected protocols = 1,3,5
logical adapters 1, 3 and 5

Selected protocols = *, wan
all logical adapters

Default value of adapters specification has form:
Selected protocols = *
which means all adapters except for those bound with WAN – type links

8.3.3. Time Synchronization

ASLINK network module can synchronize its time with other stations. Time synchronization with other stations can be turned off by setting the value of ***Do not synchronize time*** parameter to *Yes*. If the difference of time between the station and a remote server exceeds value set by ***Difference between server time and time of local station which is exceeded will causes delay in synchronization*** parameter, in at least as many following received packets, as defined by ***Number of successive time packets which local station has to receive from single server to synchronize its clock*** parameter, the time of the local station will be set. Maximum time difference is given in seconds. The ***Time synchronization limitations*** parameter is used for limiting the set of timeservers that the time may be synchronized with. Its value is the list of timeservers. Like in case of filters, server names can contain '?' and '*' characters. In order to the station to change its time it must receive from one time server at least the number of following time packets, equal to this defined with ***Number of successive time packets which local station has to receive from single server to...*** parameter.

In order to the station to act as a timeserver in the network it is necessary to set ***Time server settings/Server priority*** and ***Time server settings/time interval of broadcast of packets containing actual time*** parameters. If one of these values is less than zero, the station will not function as the timeserver. If the value of parameter equals to zero, value 5 for priority and 10 for interval is accepted. The server priority is the number that specifies timeserver importance. In the specific moment, there is only active one server with highest priority. Active timeserver is broadcasting network packets

containing current time of the timeserver at interval set with ***Time server settings/time interval…*** parameter. The interval is given in seconds.

If the server stops being active, its function may be taken over by another server, until this time inactive due to low priority. The station may begin functioning as the timeserver, if current timeserver doesn't send any packet during the period of time equal to the product of ***Condition for time server activation*** parameter value (defined on local station) and prevailing timeserver interval value (defined on remote timeserver). To avoid situation, when incorrect parameterization of remote timeserver (i.e. interval of 1 hour) is causing long interruptions in network synchronization, the ***Time server settings/maximal time interval of broadcast of packets containing actual time*** parameter has been introduced. If the interval of timeserver, from which the new server is not getting time packets, is greater than the value defined by this parameter, value defined by the parameter is accepted to compute period of time, after which the new server starts to work.

If the difference between server time and local time is greater than value defined by ***Long time difference*** parameter, change of the station time will yet occur after receiving at least as many packets, as it is defined by ***Long time difference*** (counter) parameter. Those parameters give the time to correct an incorrect time setting done by remote station operator.

Server sends time packets to all logical adapters used. If the timeserver's client supports a few logical adapters, it can send time packets received from one adapter to the rest of adapters. This function may be used for time synchronization in subnets different than subnet the timeserver works in. To enable broadcasting time to other networks the ***Time packets replication*** parameter should be set up on a station connected to multiple subnets. Time packets can be subject to further replication on following stations until the number of replications exceeds the value of ***Number of allowed replications of time packet*** parameter (set up for server). The parameter is defined on timeserver. To change the maximum number of replications on replicating stations the ***Number of allowed replications of time packet*** parameter should be set up for client.

8.3.4. Protection with Password

With help of AslView program it is possible to perform functions, which may have significant influence on operation of the network module. To avoid accidental or unauthorized access to these functions, possibility of protecting the network module with password has been added. The password can be set with ***Changes lock*** parameter. Its value can takes value of *Yes*, *No* or the password text. If it's set to *Yes*, access to the protected functions will not be possible. If the parameter's value is *No* the access to network module is unlimited. Other values are interpreted as password that has to be entered by user before access to password-protected function. Entered password remains valid for 10 minutes. Any execution of password-protected function causes starting the countdown from the beginning.

8.3.5. Connection to Previous ASLINK Version Workstations

Station operating under control of network module version 5.00.000 or higher can be connected to older versions of the network module only when the value of *ASLINK4* parameter is set to *Yes*. This possibility is enforces higher network load in phase of making connections. The default value of *ASLINK4* parameter is *Yes*, but in future versions of the network module the default value can change.

REMARK *After starting the network module of version lower than 5.00.000 on the workstation, previously running a newer version of the module, all remote stations with higher versions of the network module should be restarted (it is necessary to restart applications using ASLINK network module). The necessity of restart doesn't concern stations running older version of the module.*

8.3.6. Diagnostics

Parameters control the diagnostic information output by ASLINK network module. This document doesn't contain data that would allow full interpretation of diagnostic information. The purpose of diagnostics is to help in determining reasons of improper operation of the network links of **asix** system and it should be run by ASKOM employees or with their support.

Diagnostic information generated by network module can be written to the log file and/or transmitted to AslView programs connected to the network module.

The ***Log file*** parameter defines the name and location of file the diagnostic information will be logged to. Next to the name, after a comma, maximum size of the log file in megabytes can be specified.

9. Designer – Designing the Application

The basic part of each application is a set of visual masks. They are used for monitoring on the screen the current state of industrial process under control. Each mask is composed of a set of components, which task is to present the values of process variables in graphic form (collected by means of ASMEN manager) as well as the archive data (collected by ASPAD module). AS32 program allow editing masks regardless of the mode in which is executed. It is also possible to pass to design mode when operating in application mode, which allows editing the masks at run-time.

During designing the application it is possible to declare some parameters determining the way of application operation. Those parameters need to be set up in the XML application configuration file with use of Architect module.

9.1. Designer Window

Designer window is the main element of AS32 program during operation in designer mode. This window is automatically open when the AS32 program is started in designer mode or in the moment of passing to designer during operation in application mode (using a command MASKS.DESIGNER from Control Panel window).

Designer Window fulfils the function as follows:
provides set of commands used to design the application
provides control fields used for creation the masks,
monitors and stores the messages related to internal operation of the system.

Designer Window remains open during the entire period of operation in designer mode. Closing the window denotes the end of programs operation. In case where designer was started at application mode, a possibility exists to lock the operation of closing the window or its protection by means of a password.

9.1.1. Components of Designer Window

The Designer Window is presented below:

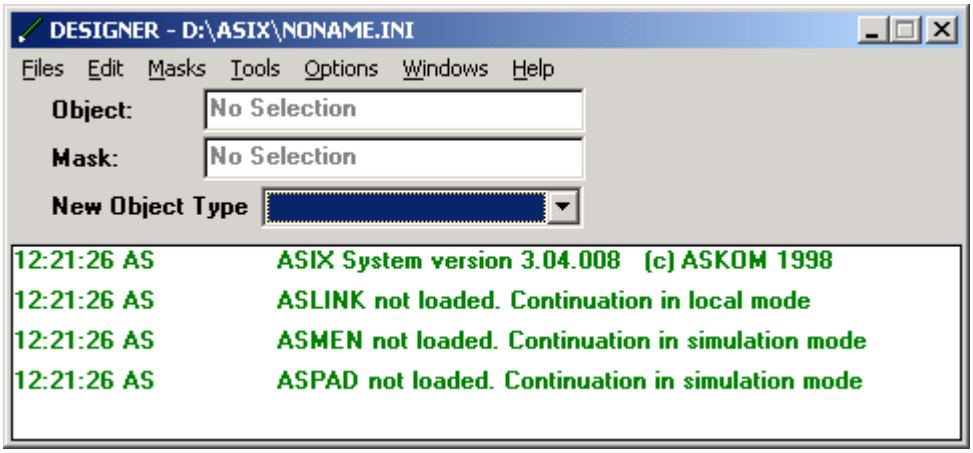


Figure 9-1. The Designer Window.

The Designer Window includes four basic elements:

header bar - the name of initialization file, which describes the present application , is specified here;

drop-down menu - the menu which opens the commands used to design applications; some of these commands are doubled by keyboard shortcuts, pressing of which gives the same effect as execution of command;

status field panel - fields useful during designer phase are located in this area:

- *Object* - the field, which specifies the name of presently selected object or the number of selected objects;
- *Mask* - the field specifying the name of current mask; if mask is in refreshing mode - so the suitable comment is shown; if mask was changed the character * appears before the mask's name;
- *New Object Type* - the combobox showing the name of object class, which shall be inserted on the mask edited mask; the field is also used to select the class, by selecting any available class from drop-down list;

list of messages - the list, in which the messages are displayed, connected with operation of **asix** system.

9.1.2. Description of Designer Window

FILES
MASKS
OPTIONS
HELP
EDIT
TOOLS
WINDOWS

Below the commands included in menu of Designer Window has been described. More detail information concerning some of these commands may found in respective topic sections. The set of available commands is different and depends on the mode of program start-up. For instance, if you start in application mode, the commands, which permit changing the application, are disabled.

FILES.NEW
Creates new initialization file, i.e. a description of a new application. Some settings may be retrieved from recently edited application.

FILES.OPEN
Opens the existing initialization file. This denotes passing to editing the application, which had been created previously. You should bear in mind that visual masks are not connected directly with any initialization file. The same masks may be used in various applications, under condition that mask paths as well as the definitions of fonts and bit maps are set as required.

FILES.SAVE
Saves the present state of initialization file on disk.

FILES. SAVE AS
Saves the present state of initialization file in a new disk file with a new name. It corresponds to creation of new application with settings identical with the current application.

FILES. CONTROL PANEL
Passes to execution mode of application. This command is available only AS32 program was started in application mode.

FILES.HIDE WINDOWS
This command hides all open windows and on system bar **asix** icon is displayed. To restore windows double-click on this icon.

FILES.EXIT
Closes the Designer Window; it has the same effect as completion of program operation. If designer was started within the range of application system then this command may be locked or protected with a password by declaring the ***Exit lock*** item in the application configuration file (parameter set up with use of Architect module: Architect > *Fields and Computers* > *Security* module > *Locks* tab).

EDIT
drop-down menu, which contains commands related to edition of objects on visual masks. This menu shall be described in the section related to editing masks.

MASKS
Drop-down menu, which contains the commands related to operation on visual masks. This menu shall be described in the section related to editing masks.

TOOLS.KEY_SETS
Opens the window used to manage the files, which contain descriptions of sets of keys. Sets of keys enable to define the method of keyboard usage while application is executed.

TOOLS.FILE_MANAGER
Opens the window, which enables execution of series of basic operations connected with files management, for instance: copying of files, deleting of files. *'Files Manger'* window is presented below:

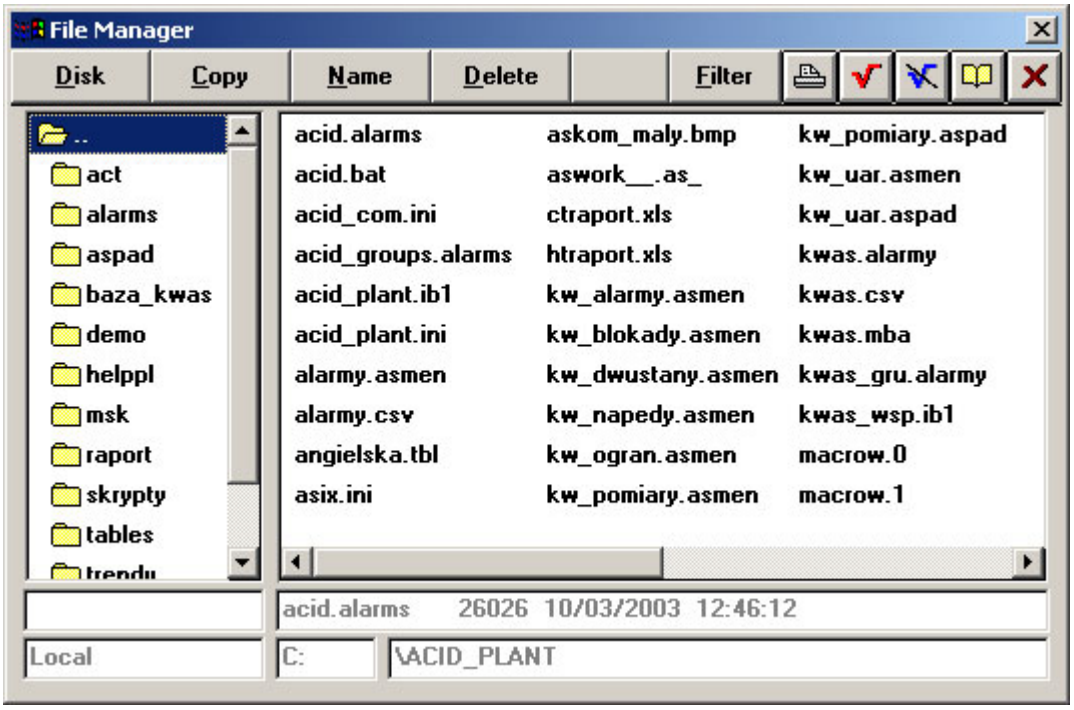


Figure 9-2. 'File Manager' Window.

Disk	- the command enables to change the disk drive that keeps files subjected to operations of the File Manager. The command allows also moving to another network station.
Copy	- the command used to copy the contents of files. Both copying within one station and transfer of files to and from another network station are possible.
Name	- the command is used to change the filename.
Delete	- the command is used to delete all the permanently selected files or the current file.
Filter	- the command used to reduce the set of the shown filenames.

- click this button to display the window that makes possible printout of the selected file.

- click this button to make permanent selection of all the filenames of the list.

- click this button to deselect all the previously highlighted filenames.

- this command is to refresh of the contents of the current directory. The command should be used if there exists an opportunity that the contents of the directory could be changed while the File Manager was running, because of operating of other modules of the AS32 program (e.g. savings of the screen content) or other programs.

- click this button to close the File Manager window.

TOOLS.VARIABLES
Opens the window, which enables to browse the list of process variables (current and archive ones) defined in the system.

TOOLS.TEXT DESCRIPTION
The command provided for documentation purposes of application. It allows creating files, which contain text description of some mask parameters being part of an application.

The especially important feature of text description is possibility to obtain a cumulative description of all process variables used on masks. Execution of command causes the window to be opened.

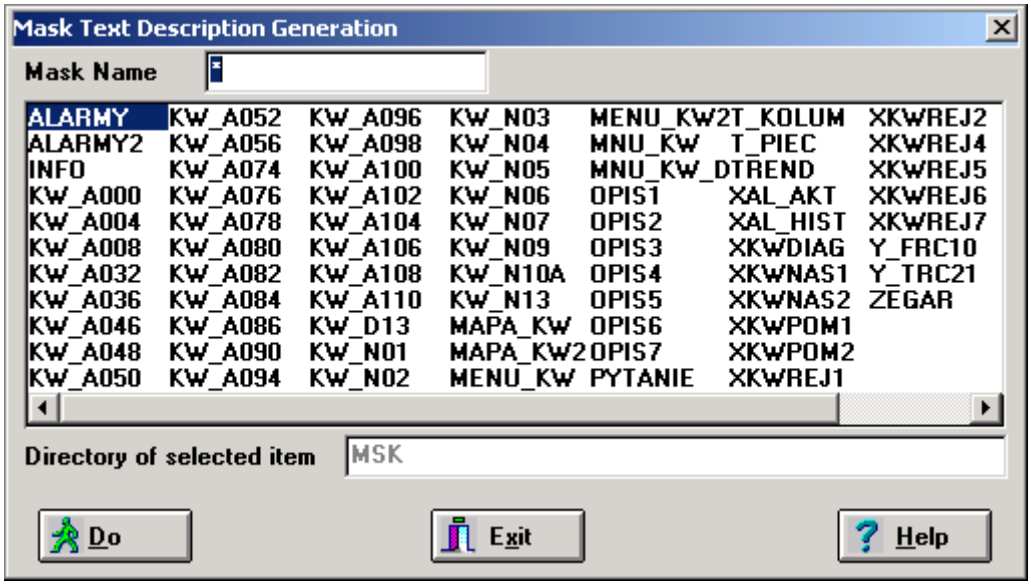


Figure 9-3. 'Mask Text Description Generation' Window.

You can generate the text description by selection of mask name and clicking on *Do* button. It is possible to obtain descriptions of many masks at the same time with one operation. To do this specify the name of group of masks in the field *Mask name* using wildcard * and ? (usage convention corresponds with naming of files). Especially, if you specify only * in the above mentioned field,

you create descriptions for all masks of application (i.e. masks included in the path).

Text descriptions are saved in files of same names as those of a mask, to which the description relates, and TXT extension.

TOOLS.GRAPHICS_PRINT
The command provided for documentation purposes of application. It allows creating graphic printouts of masks being part of application. Handling a window being open is the same as in case of window of the command TOOLS.TEXT DESCRIPTION. An additional element is requirement to declare format, in which the printout is to be output. It is possible to create the mask image using the bitmap file format or print directly on selected printer. In the later case you may omit the phase of printing a hardcopy saving the effect of printing operation in the file. The result of operation is saved in files of names identical with name of the mask and BMP extension. A graphic printout window is presented below.

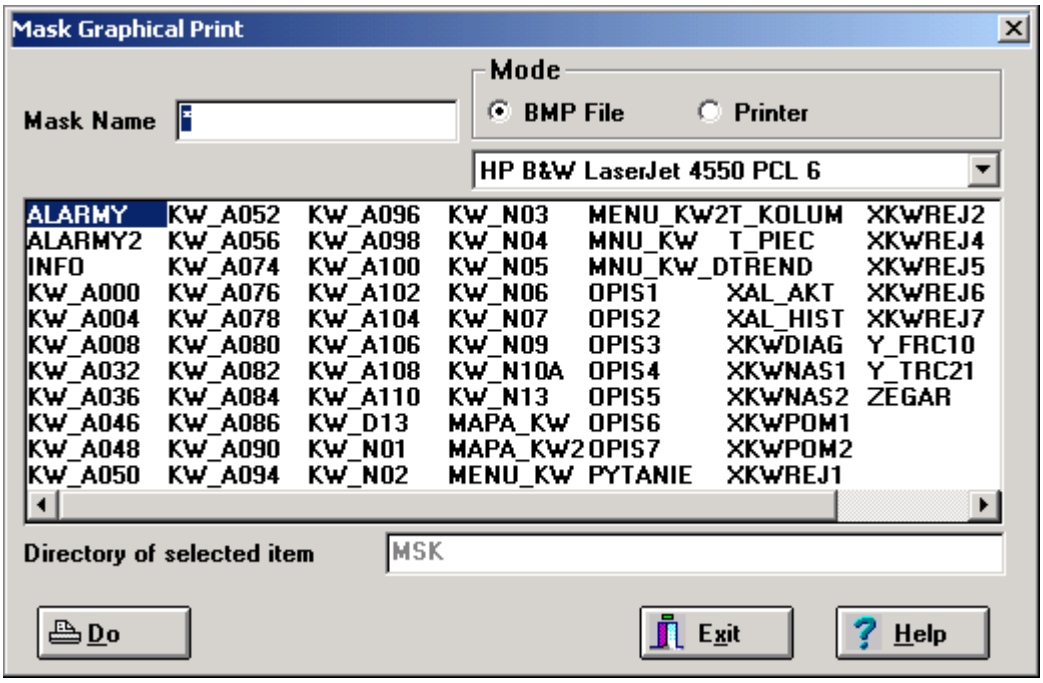


Figure 9-4. 'Mask Graphical Print' Window.

TOOLS.BITMAPS_EDITOR
Opens the bitmap for editing. Editor enables creation and modification of icons and images located in image library file used in created application.

TOOLS.TEXT_EDITOR
Opens the text editor window. This is a simple editor, which allows modifying any text files in ASCII format. It should not be used for text edition of a current initialization file (AS program saves its own copy of content of this file, destroying changes entered during edition).

TOOLS.REPORTS_WINDOW
Opens the window of report system, which allows generation of selected reports as well as other operations related to reports.

TOOLS.DIALOGS_WINDOW
Opens the *'Dialogs Window'*, which enables interactive calculation of arithmetical expressions, which use values of process variables.

TOOLS.ACTION_SCHEDULER.TIMES

TOOLS.ACTION_SCHEDULER.EVENTS
Opens the window used to declare the periodically executed actions or fulfilling defined conditions.

TOOLS.PRINTER_MANAGER
Opens the window used for handling a printer. This window is provided for monitoring the presently executed printouts, management of printout queue as well as execution of files printing.

TOOLS.PASSWORD_MANAGER
Opens the window used for handling the passwords. This window is provided to set the validity condition of passwords as well as for changing the content of passwords

TOOLS.USER_MANAGER
Opens the window enabling creation a new user (when logon system is used).

TOOLS.VARIABLES_TABLE
Opens the window for monitoring the values of process variables.

OPTIONS.TIME_CHANGE
Opens the window that allows changing the time.

OPTIONS.VIEW_POSITION
Enabling the window of mouse position during edition of objects on masks.

OPTIONS.CURSOR_SNAP
Sets the method of cursor movement during edition of objects on masks.

OPTIONS.OBJECTS_EDIT
Changes the mode of edition of objects. This command is provided for selection of type of dialogue window used to change the object definition.

OPTIONS.DEMO
Recording or playing the demo sequence.

WINDOWS
List of opened windows.

HELP.INDEX
Displaying the index window of online help.

HELP.ABOUT
Displaying the window with system information, which contains a lot of auxiliary information.

9.2. Creation of Synoptic Masks

The basic part of each application of **asix** system is a set of masks used for monitoring of industrial process. In the system, the masks to handle the alarms as well as the synoptic masks are distinguished, which are used for graphical monitoring the values of process variables. The synoptic masks may be also used to carry-out the control operations.

9.2.1. Mask Definition Window

Opening Mode
Components
Operations

The basic commands used to develop and modify the masks being a part of application are provided in menu MASKS of Designer Window. New Mask is created by means of command MASKS.NEW_MASK.SCHEMATIC. This opens the below dialog window provided for defining the basic mask parameters.

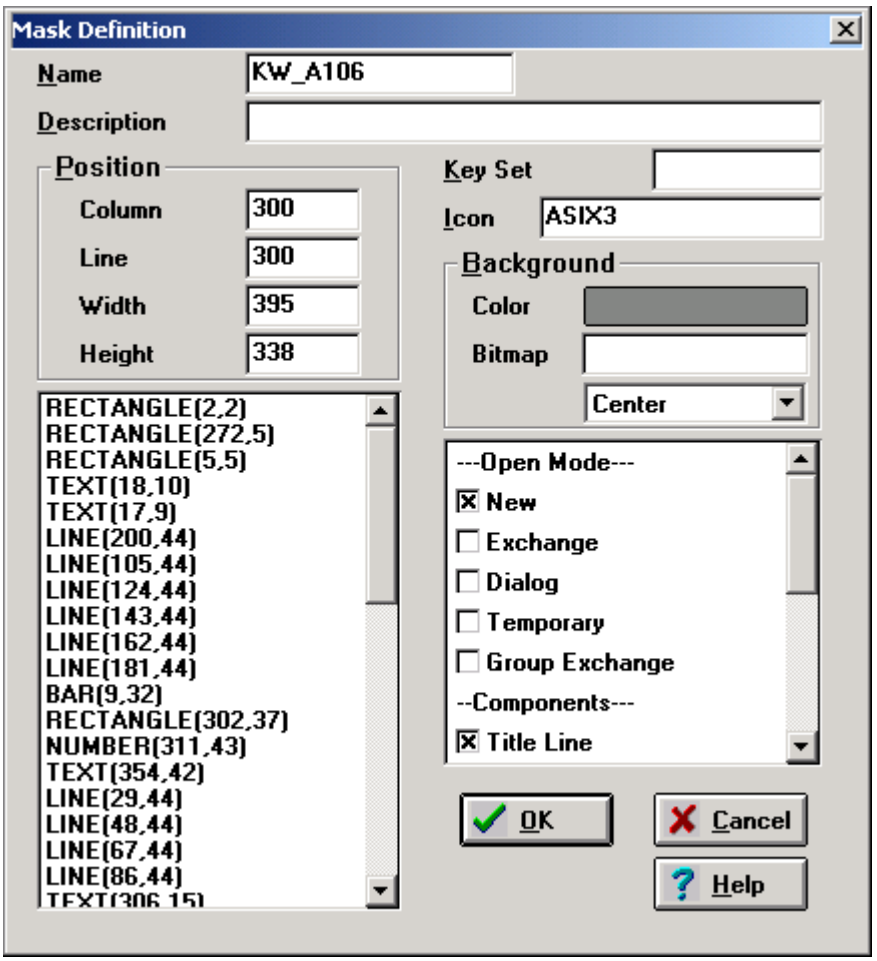


Figure 9-5. 'Mask Definition' Window.

In the field *Name* the designer should give maximum 8-character symbolic name of a mask. This name shall be used in the system to identify a mask. Set of characters, which may be used in mask name is the same as used for names of files. It is due to the fact that symbolic name of the mask is also a name of disk file, in which the mask contents is stored (the MSK extension is added automatically).

The field *Description* is provided for text to be displayed in line of mask title. If the mask has no title line then the *Description* field may be considered as a pure document field provided for description of mask destination.

Group of edition fields *Position* is used to define the position and size of a mask. Mask co-ordinates may also be changed later (when a mask is displayed) with a mouse. The designer may move a mask „catching" it by title line or to pull out a frame of mask. In the case of large mask with scrollbars the real size is calculated on the basis of object location. Increase of mask area can be obtained by moving any object beyond the mask.

In the field *Key set* you may specify (not obligatory) the name of a keyboard shortcuts associated with a mask. You may choose the name of a shortcut in the selection window, which is open by means of *Grey* + key or by clicking the right mouse button in the field. The detailed rules of using a keyboard shortcuts are presented in a separate section (*See: 9.3.2. Defining Keyboard Shortcuts*).

The button *Background-Color* is provided to specify the background color of mask. Clicking the button with mouse you open of color window, where you may select (by indication) required color.

The *Background-Bitmap* button allows defining a bitmap, which will be displayed as a background for the synoptic mask. Clicking it with right mouse button you open the selection window, where the desired bitmap in given application's bitmap file can be indicated. Also, the name of the bitmap with its extension can be specified. File of this bitmap should be included in mask directory.

In the field *Icon* you have to specify the name of an icon, which shall be displayed after operation of mask minimizing. You may also select the name of icon in selection window, which can be opened by means of *Grey* + or by means of clicking the right button within the area of a field.

In the list placed on the left side of the window, names of visual objects being the part of the mask will be later inserted. Of course, this list is empty just after the mask was created. List of objects permits execution of some operations connected with handling objects. They will be described later in this section.

The list on the right side of the window is used to define various attributes of mask being created. You should set check boxes included in the list in accordance with required parameters of mask.

Meaning of individual fields within the list of attributes is as follows:

Open mode - declares the default mode of opening of a mask; synoptic masks are typically open with use of OPEN_MASK operator action; opening mode may be given as a parameter of this action; if

	it is not be specified the mask is open in mode specified in definition of a window;
New	- opening the mask will have no influence on current mask in the moment when the operation is executed;
Exchange	- opening the mask will close the current mask in the moment when this operation is executed;
Dialog	- opening of mask will have no influence on current mask in the moment the operation is executed; after mask is opened there will be no possibility to pass to other mask until mask, which has been opened in *Dialog* mode is closed;
Temporary	- opening the mask will have no influence on current mask in the moment when the operation is executed; after mask is opened, each action e.g. clicking other mask or a window will cause automatic closing the temporary mask;
Group Exchange	- opening of mask will close of all masks not being locked (masks which have permission to be closed);
Components	- declaration of components of a mask;
Title line	- mask will have title line;
Border	- mask will have standard frame;
System button	- mask will have system menu button. system menu permits executing operations such as moving a mask, closing a mask, etc;
Operations	- declaration of set of permissible operations executed on mask;
No move	- mask will not be moved (in application mode);
No size	- size of mask will not be changed (in application mode);
Maximize	- mask maximizing will be possible; selection of this field will cause that maximizing button will be added to a mask;
Minimize	- this will permit minimizing a mask to an icon; selection of this field will cause that the minimizing button will be added to a mask;
No close	- a mask will not be closed;
Always on top	- a mask will always be seen (state of *Always on top* masks is dynamically switched off at the moment of selecting the other application; it allows to use other programs even when the **asix** application has masks opened in the *Always on top* mode).
Scaling	- an attribute set during mask configuration; it activates scaling of object size and position when mask size is changed in application execution mode. In Designer mode a mask may be calibrated by pressing the *Alt* button during resizing it, apart from the state of the *Scaling* parameter.

After you enter the parameters of mask and click on *OK* button , new mask will appear on the screen. From this moment you may add visual objects to a mask. You may return to the window of basic parameters at any moment. Click twice the left mouse button within the window area (beyond object boundaries) or execute the command MASKS.CHANGE (if no object is selected).

9.2.2. Mask Selection Window

If change of definition of a mask created earlier is required (changing of basic parameter or objects definition), you should execute the command MASKS. OPEN. The below dialog window will be opened presenting the list of all masks available in the application.

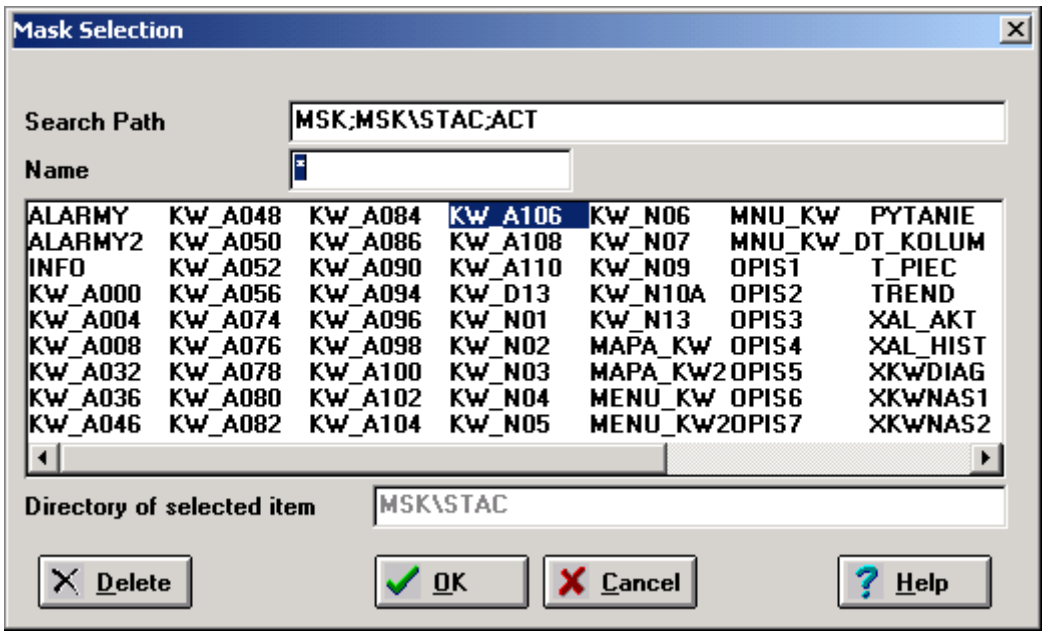


Figure 9-6. 'Mask Selection' Window.

Visual mask is available in application when its definition file is included in one of directories specified in field *Search path*. Path is defined as a list of directory names separated with semicolons. The path is created while initialization file is developed. The path may be freely changed in masks selection window that permits saving the application mask in various directories. However, you should remember that search path is common one for all elements of application: visual masks, keyboard shortcuts and compound actions. Change of path in mask selection window has influence on the access to keyboard shortcuts and actions.

You open a mask by selecting the mask name in the list and clicking on *OK* button. However, as an alternative, the name of selected mask may be entered manually in the *Name* field. This field may be also used to limit the number of masks on the list. Due to proper usage of wildcard * and ? characters you may restrict the number of presented masks to selected sub-set. The field *Directory of selected item* is an additional field, which shows in which directory a file defining a mask chosen on the list is included.

Mask selection window may also be used to delete masks, which are not longer necessary.

Operation of copying a mask may be executed with use of two methods. The first one consists in usual copying the mask definition file (for instance, using the Windows Explorer). While copying a mask definition file you should bear in mind that file name is the same as symbolic name of a mask and that *MSK* extension cannot be changed. The second method of copying the masks (without leaving the AS program) consists in opening a selected mask by means of command MASKS.OPEN, entering the window of basic parameters (double clicking on the mask or a command MASKS.CHANGE) and then changing the symbolic name. In the moment of closing the window the AS32 program will ask you if it has to change the name of existing mask or if it has to create a new mask of given name and of the same definition as the original mask.

9.2.3. Description of MASKS Menu Commands

The remaining commands of menu MASKS are described below. They operate usually on a mask, which is currently selected. Name of this mask is shown in *Mask* field in Designer Window. These commands relate also to alarm masks.

MASKS.STORE
Saves the present mask definition in disk file. To prevent against loss of our effort (for instance because of power blackout) this command should be periodically executed. The command is doubled with *Alt-F2* keyboard shortcut.

MASKS. CLOSE
Close editing the mask. It will be removed from the screen. If the mask was modified, program asks a designer whether changes are to be saved in a file.

MASKS.CHANGE
Opens the basic mask parameters window or defining window of visual object, if an object is selected (condition of selection of object is given in the field *Object* in Designer Window)

MASKS. REFRESH_MASK
The command causes switching over a mask into refreshing mode. If AS program was started in application mode, this denotes that the mask is used for normal managing of a process. In case of work exclusively in designing mode, mask refreshing permits the general test of mask operation on test data (although access for real data may be obtained, if parameters of ASMEN communication manager had been properly set).

NOTICE When REFRESH_MASK is stopped in the Designer mode, data archiving is also stopped.

MASKS. EDIT_MASK
This command causes passing a mask from refreshing mode to edition mode.

MASKS. EDIT ALL
This command disables refreshing mode for all currently opened visual masks.

9.2.4. Context-Sensitive Menu for Mask Edition

The possibility of using context-sensitive menu is a real facilitation when editing masks.

The menu consists of commands of the *Edit* menu, several commands of the *Masks* menu and commands for object/pattern selection.

The menu form changes depending on the number of chosen objects, defined patterns, etc. There are only the commands that can be performed in a current time available.

The context-sensitive menu is activated by clicking the right mouse button within the edited mask area.

9.2.5. Object Edition

Inserting New Object
Object Selection
Changing Object Parameters
Changing Position and Size
`Undo' Command for Operations Changing Position and Size
Objects Deleting
Object Copying
Setting Object Order
Grouping Objects
Patterns of Objects
Hiding Objects
Auxiliary Operations
EDIT.ALIGN_TO
EDIT.JUSTIFY
EDIT.PUT_ON
EDIT.SYMETRIC_TURN
EDIT.MOVE NEAR
EDIT.LOCK
EDIT.UNLOCK

After basic mask parameters are established – you may start edition of visual objects. Objects are the basic part of each mask. They are provided for graphic presentation of a process under control. Because of their features, objects may be divided into following categories:

static - static objects, which are always displayed in the same way. They do not use any process data. They are only designed to present proper graphic form on a mask.

dynamic - variable dynamic objects, whose appearance depends on data connected to them (basically, ASMEN process data or ASPAD archived variables)

control - objects, which enable executing operations of industrial process control. Control process is based on dialog with process operator, which can sent appropriate values with use of of object to the controller. Control objects usually belong also to dynamic objects category.

selectable - objects, which have the ability to react on the action of the operator initiated by the mouse or

	the keyboard. As a result of this reaction, object can execute control operations.
special	- objects, which are not directly designed for process monitoring. Their task is to execute diagnostic or service functions (for example calculating variables in object, class *CALCULATOR*).

Categories of objects don't have to be separable. For example, static object cannot be a dynamic one at the same time, but for example dynamic object can be control object and (or) selectable one at the same time.

Except the conventional division into categories, there is also object division into classes. The object, which belongs to certain class, has strict rules of behavior (displaying, access to data, way of carrying on dialogue). Creating a synoptic mask means to insert on the mask proper set of objects. It is recommended to use object of these classes, which meet the requirements set upon the mask during designing phase. Each object also has individually declared parameters, which define its specific features (general rule of behaving of an object always depends on its class).

Inserting a New Object

Inserting a new object on the synoptic mask consist of two steps. In the first one you have to specify the class of the object being created. To help choosing the class, there is a drop-down menu *New Object Type* in designer window. On drop-down menu you have to select the name of the object class. Then transition to inserting object mode will take place indicated in the field *New Object Type* by the name of chosen class. The next step is to specify the location and size of the object. The way of making this operation depends on the class of the object. Used methods of specifying co-ordinates are as follows:

single point	- click with left button in the right place; the size of the object is set automatically;
rectangle	- click the left mouse button to indicate the left corner of area of the object, you then move the mouse into place of the second corner and release the mouse button;
line	- click the left mouse button to indicate the starting point of the line, you then have to move the mouse into place of the second point and release the mouse button;
polyline	- you would define successive lines in the same way as above; starting point of next line is always at the end of its preceding one; clicking the right mouse button indicates end of defining the polyline;
text	- click the left button sets the cursor; next you have to enter text, which will be displayed by the object; pressing *Enter* key completes entering of the text.

After specifying the location and size, the new object is created. Its detailed parameters are set by default. As a rule they are similar to recently inserted object of the same class. All parameters of the object, its size and location can be changed later.

After creating an object, name of the class is removed from the field *Type of new*. It means, that program returned to edition mode of existing objects. Inserting the new object requires new class to be chosen. If the object is to be the same class as the last one, you only need to click the right mouse button in the area of the mask. This will cause a change into inserting mode. Clicking the right mouse button can be also used to leave the inserting mode without creating a new object.

Object Selection

Each operation, which is to be performed on the object, requires its earlier selection. The object being selected is surrounded with a frame. Simultaneously its name appears in the field *Object* in designer window. Some of the editing operations are performed on the whole group of objects. You should carry out then simultaneous selection of all objects, on which the operation is to be performed. Each selected object will be surrounded by a frame and an information showing how many objects are selected will appear in the field *Object*.

In the simplest case, clicking with the left mouse button in the area of the object select a single object. Such operation automatically unselect previously selected object. But, if during selection of the object, the *Alt* key is pressed, the previous selection is kept. This allows performing repeated selections. If you want to unselect one object but keeping the conditions of the object, you should do it similarly: holding *Alt* key, click on the right object. Clicking on a given object does not change selection states of any other objects. In general, holding *Alt* key means, that you shouldn't change the selections of any other objects. Selecting a single object can be carried out also by means of keyboard. Pressing *Tab* and *Shift-Tab* keys causes transition via successive objects (in sequence of their order on the mask). The *Alt-(grey)** keyboard shortcuts selects all objects on the mask.

In case of multiple selections, it is easier to use selection of area. To select area click the left mouse button anywhere on the mask's area, than keeping button down move the mouse and release button. The rectangle drawn by mouse coursor defines the selecting area. Every object included in selecting area will be selected. All object covered partialy by the selecting area is also selected.

Changing Object Parameters

Parameters, which are specific for given object may be changed with use of special dialog windows. The object dialog window is opened by means of one of the following operations:

- double-clicking with left mouse button within the area of the object,
- performing MASKS.CHANGE command after object selection.
- pressing Enter key after object selection,
- choosing proper object in list of objects in mask definition window, then pressing Enter key or double-clicking with left mouse button.

There are three types of definition windows. What window is open, results from the recently performed command OPTIONS.OBJECTS_EDIT. Types of definition windows are as follows:

Full definition - window containing all parameters of the object (except position and size). Form of the window depends on class of the edited object

Process variables - window used to monitor and to change names of ASMEN process variables, which the edited object is using. This type of window is available only for objects using ASMEN variables. The form of this window is the same for all object classes.

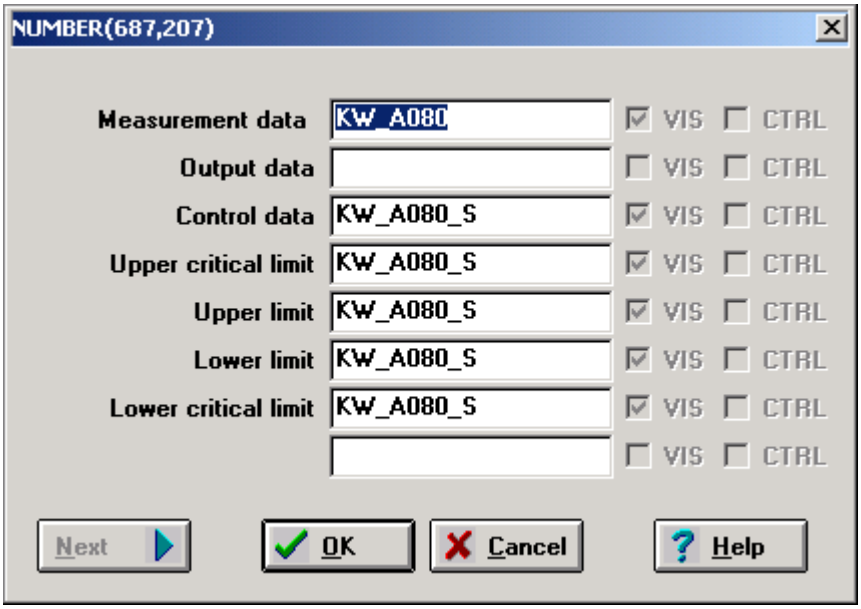


Figure 9-7. The 'NUMBER' Window for Process Variables Edition.

In the window there are names of all process variables used in object. Descriptions of fields depend on the class of the object and they describe usage of variables. Field *VIS* informs whether process data is being taken from the process. Field *CTRL* informs whether the data is used to control the process. If the object uses more than 8 variables, the key *Next* is used to browse the variable list.

Coordinates - window, which enables changing object's position and size. Its shape is the same for all object classes. Some classes of objects do not allow changing their size.

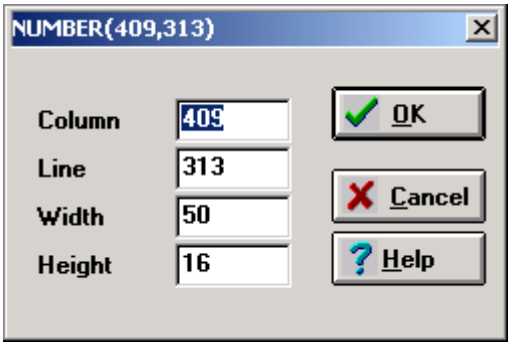


Figure 9-8. The 'NUMBER' Window for Coordinates Edition.

Coordinates window is a supporting mechanism. As a rule mouse is used, directly on the area of created mask. Coordinates window opening can be done in another way by double-click on the object together with holding *CTRL* key at the same time. A coordinates window is being opened then or when editing was switched permanently to coordinates - the full definition window.

Changing Position and Size

Basic method of changing the object coordinates is moving them with use of mouse. To move the object click with left mouse button in the central part of object and drag it to a new place and release the key. Size changing is carried out in the same way. The only difference is the place of clicking. You have to do it near that edge (corner) of object area, which is to be moved. Some object classes do not allow changing the size or allow moving only some of its edges.

Moving the group of objects is also possible. It allows moving simultaneously some objects, with their relative position maintained. Group moving must be preceded by selection of objects, which takes part in this operation. Next you have to start moving any of the chosen objects, holding *Alt* key pressed. After beginning the move, *Alt* key does not have to be pressed. End of the operation is carried out in common way. In effect, all selected objects are moved in such way, to keep relative position in relation to moved object. While moving many objects, frames of all selected objects are displayed.

In all selecting operations, size and position changing, you may use the keyboard (especially in case of necessity of very precise positioning). Operation is started with use of mouse, next the arrow keys are used (we should always remember to hold down mouse button). Every pressing down the arrow key causes movement of cursor of one unit (step) in proper direction (under usual condition one unit is equal to one pixel). Operation is ended when mouse button is released or when *Enter* key is pressed. Operation can also be abandoned at any moment by pressing *Esc* key.

Monitoring of the mouse cursor position is very helpful, while of moving and size changing operation. By means of OPTIONS.VIEW_POSITION command you can open a small window, in which the coordinates of current mouse position are displayed. Mouse position is updated only, when cursor is inside of the mask. The upper left corner coordinates or the object size is displayed in the monitoring window while moving of the object or resizing operation are performed respectively.

In normal work mode mouse cursor is moving through the area of mask with one pixel step. There is a possibility to define a bigger step of the cursor. Increased jump helps to place objects more equally, for example in columns. Cursor step setting is carried out by OPTIONS.CURSOR_SNAP command, which causes displaying such dialogue window as shown below.

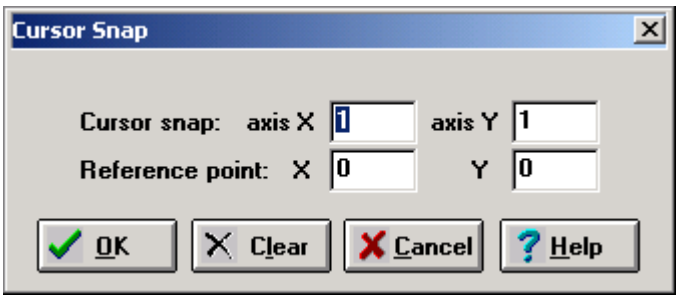


Figure 9-9. The Window for Cursor Snap Setting.

It is possible to set independently the step of cursor in both coordinates. The value of the step is expressed in pixels. Definition of reference point allows moving the point, where the mouse cursor will be positioned in. For example, if the step is (10,20) and the reference point is in position (5,8), the cursor can be positioned in columns 5,15,25... and lines 8,28,48...only.

'Undo' Command for Operations Changing Position and Size

EDIT.UNDO_MOVE command is available for the sequence of operations that change position and size of objects. The mentioned operations can be cancelled with use of 'Undo' command till the designer changes state of object/mask.

You can use the command also by pressing *Ctrl-Z* key.

Object Deleting

The object placed on the mask may be deleted from it. This is carried out by means of EDIT.DELETE command after previous object selection. You can delete several objects at the same time by pressing *Ctrl-U*, *Ctrl-Q* or *Del* key.

If object(s) was deleted by mistake you can undo this operation with using the EDIT.UNDELETE command. Only objects, which were deleted in last deleting operation, can be recovered.

You can also delete objects in basic mask parameters window. To do it you have to select the proper object and then press *Ctrl-Del* shortcut.

Object Copying

A very common situation during mask designing is a need to create a lot of nearly the same objects. The easiest way to do it is to use EDIT.REPEAT_OBJECT command (doubled by *Ctrl-P* and *Ctrl-R* keyboard shortcuts). This command creates the exact copy of selected object(s). Created objects are only a little displaced in relation to their original copies. All you need to do is to move them into proper location and to perform necessary definition modifications (for example changes of the variable names).

Created object copies remain selected. This allows creating lot of copies, by pressing *Ctrl-P* or *Ctrl-R* keyboard shortcuts several times. In case of copying several objects, an instant group dislocation is possible.

Copying objects between masks is also possible. To do this, you have to open two synoptic masks to be edited. On one of them choose the objects to be copied, next perform EDIT.CUT command (*Ctrl-T, Ctrl-X, Shift-Del*) or EDIT.COPY command (*Ctrl-K, Ctrl-C, Ctrl-Ins*). Both of the above mentioned command copy indicated objects to the buffer. In addition, CUT command, deletes the objects from the source mask. Next you have to perform EDIT.PASTE command (doubled by *Ctrl-W, Ctrl-V, Shift-Ins*). As a result, in the *New Object Type* field *Paste* text will appear indicating object inserting mode from the buffer. Now you have to point the position in the destined mask, where the object inserting is to take place on. Inserting operation can be repeated lots of times (copying the same object lots of times).

Setting Object Order

Objects, which are included in the mask, are at first in the same order as they were created. Changing this order may be necessary in the following cases:

- In phase of application executing, operator can control the process by means of control objects. If the control is executed using the keyboard, Tab and Shift-Tab keys enable you to move through the objects. The order of movement depends on object sequence. Mask designer should ensure, that the objects are arranged in the way, which allows to move through the objects in the desired way.
- In case where objects cover each other, order of displaying them is important.If a big object entirely covers other object area, selecting these objects using mouse becomes more difficult. Moving a big object (even for a moment) to the end of the list can be very helpful.

AS program makes accessible two methods of changing the object order. First method consists in using the object list, which can be found, on the mask basic parameter window. This list allows the entire control of objects order. Object displacing consists of indicating the object and then moving it into desired location using *Ctrl-Home, Ctrl-End, Ctrl-arrow_up, Ctrl-arrow_down* keyboard shortcuts.

The second method of order changing consists of using EDIT. MOVE ON TOP (*Ctrl-Home*) and EDIT.MOVE TO BOTTOM (*Ctrl-End*) commands. These commands are very helpful in cases where objects are covering each other. EDIT. MOVE_ON_TOP command causes inserting before the first object, whose area is covering the area of selected object. EDIT.MOVE_TO_BOTTOM command causes inserting behind the last object, whose area is imposed with the area of selected object.

The EDIT. MOVE ON TOP and EDIT.MOVE TO BOTTOM may be used simultaneously for many objects.

Auxiliary Operations

In EDIT menu of designer window, there is lots of auxiliary operations, which are used for object editing. As a rule, they are used to operate on group of objects. Meaning of these commands is as follows:

EDIT.ALIGN_TO
This command allows to line-up the objects. You have to select a group of objects and specify the method of aligning. Aligning is performed in relation to the extreme object in desired direction.

EDIT.JUSTIFY
Object justifying allows the uniform arrangement of objects. After the group of the objects is selected and the justifying in the desired axis is performed the objects are placed in the way ensuring the constant distance between them. The most left (upper) and right (bottom) objects remain in their positions. All other objects change their position horizontally (vertically).

EDIT.PUT_ON
This command is used to locate objects centrally one after another. As a rule, this is used in case you want to insert a dynamic object in the centre of static object background. The bigger object remains on its original position.

EDIT SYMMETRIC TURN
This command allows creating mirror-copies of object. It has sense only for some of the objects. (for example objects of POLYGON class).

EDIT.MOVE_NEAR
This command causes such object arrangement, that their edges are in contact with each other.

EDIT.LOCK
In case when some of the objects have been already properly defined and the mask is still being edited, locking editing of objects already created may turn out appropriate. This will prevent from casual changing (and necessity of making next corrections). To do this, you can use EDIT.LOCK command, which may lock the possibility of moving the object or changing its size. Copying the locked objects causes, that new objects have the locks automatically removed.

EDIT.UNLOCK
This command removes the inserted by means of EDIT.LOCK command locks.

Grouping Objects

Objects put on a mask can be grouped in a constant way. The following commands of the EDIT.GROUPS submenu are used for group management:

Group Objects	- constant grouping of selected objects;
Ungroup	- constant ungrouping;
Ungroup Temporary	- temporary ungrouping that allows to edit individual objects and to change their location; after the mask closing and reopening the objects are automatically grouped again;
Group Again	- it allows to group again temporary ungrouped objects.

A click on any object of a group causes selection of all objects of that group and a double click causes opening of the *'Process Variables Exchange'* window. There are all the variables, used in the selected objects, listed in this window. After declaring new variables, they will be entered into the proper places of the selected objects – the new variables will be assigned to those objects. Exchanging concerns variables declared in the window of object

parameterization, variables used in expressions and declared in operator actions of objects.

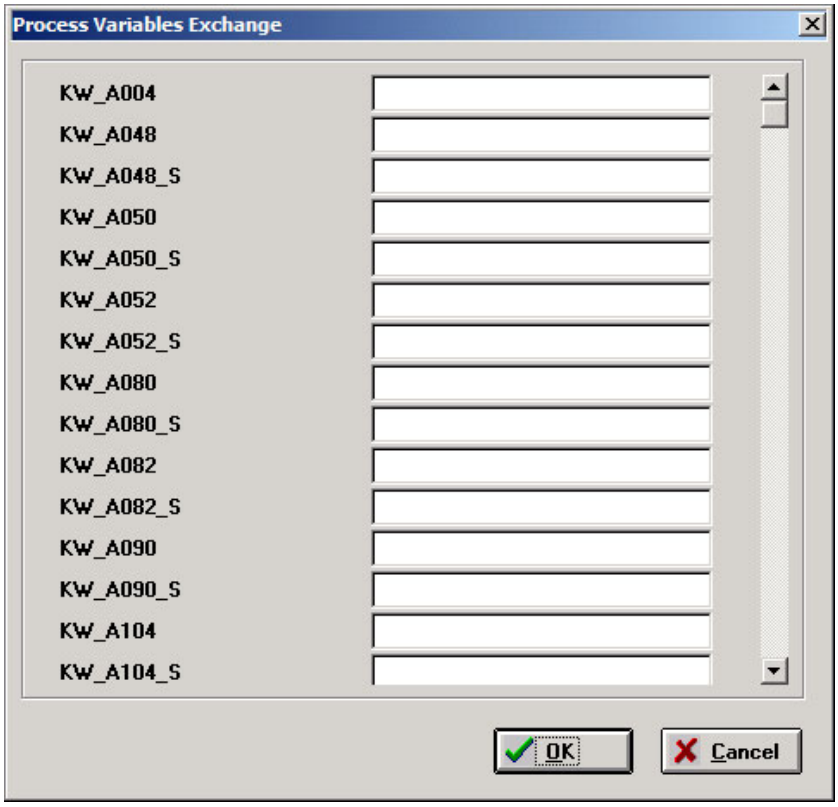


Figure 9-10. The 'Process Variables Exchange' Window.

NOTICE Edition of an individual object is possible after by a double click (with the pushed *Ctrl* button) on that object.

Patterns of Objects

Each set of objects can be stored as a pattern by means of the EDIT.GROUPS.save_as_template command. Performance of this command causes the *'Objects Template Definition'* window to be opened. This window allows to change variable names from original to more descriptive ones, declare name and description of the pattern, and write all this into the file with the MST extension.

NOTICE In this case the *'Objects Template Definition'* window causes only the names of variables, not the variables, to be changed.

All the patterns stored in .MST files can be freely copied between applications.

The pattern can be edited. The *'Mask Selection'* window allows the designer to choose among standard masks and pattern ones. Putting the pattern onto a mask is similar to putting an ordinary object. In the *New Template Type* field you can select the pattern to be used on your mask. After pointing the location the pattern will be put into, the *'Process Variables Exchange'* window will be

opened to allow you assigning new variables to the objects, and then all objects of the pattern will be put onto the mask as a group.

For the *'Process Variables Exchange'* window, exchanging concerns variables declared overtly in the window of object parameterization, variables used in expressions and declared in operator actions of objects.

NOTICE The patterns can also be created for a single object.

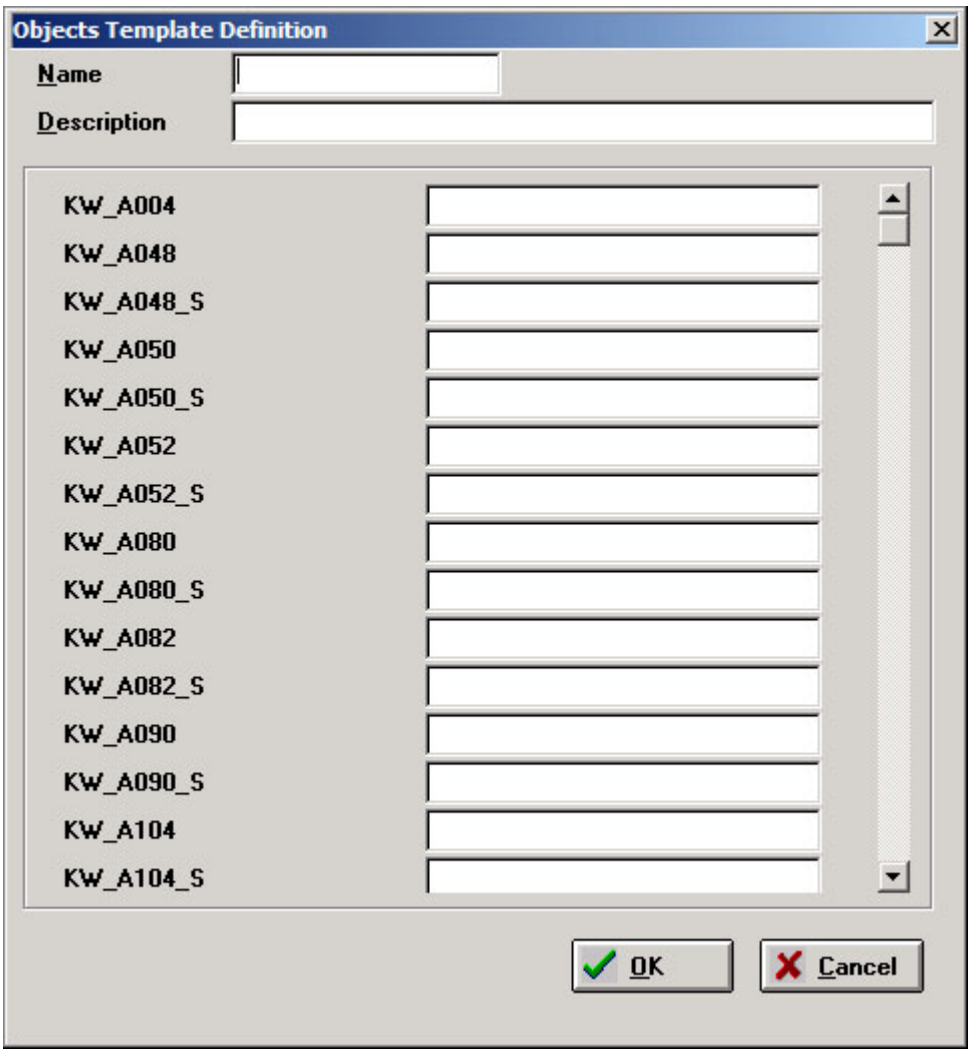


Figure 9-11. The Window for Definition the Template of Grouped Objects.

Hiding Objects

The **asix** system allows to control visibility of process mask objects. This mechanism applies to all object classes. The invisible object is not displayed and doesn't execute any functions. For example, it is possible to hide BUTTON and CALCULATOR objects in this way.

The object visibility control is not declared in a given object. It is set up with use of OBJECTS CONTROLLER object, which can control the visibility of other

objects on the basis of process variable state or the currently set password level (See: *18.4.17. PICTURE Object*).

9.2.6. Replacement of Process Variables During Mask Opening

Replacement of process variables used on the mask can be performed during mask opening. It allows to use a single mask file for opening many variants which differ in variables.

OPEN_MASK Action

The variables being replaced and their new equivalents are passed in the OPEN_MASK action, where the *mask_name* parameter has to be declared in one of the following ways:

mask_file#exchange_file

It is necessary to declare the way of variable name exchange in the .VRT file. The file format is as follows:

line 1: *text description* – if the mask has a title line, the description is put into it;
line2 to n: *source_variable_name*; *entered_variable_name*

NOTICE There is a button that allows to generate the pattern .VRT file of name replacement for the edited mask.

mask_name#source_variable_1;entered_variable_1[…#source_variable_n; entered_variable_n]

In the second way all pairs of equivalents are entered directly into the action content.

SCRIPT Action

Replacement of process variables may be also performed with use of SCRIPT operation - the part where script parameters are declared. Because the structure of SCRIPT action parameters is not fixed, the application designer has to designate places of occuring variable name by placing variable name between $ signs.

During script execution the $ signs have to be removed. Scripts displaying call parameters (without $) in Control Pannel are the following:

in VBScript:

```
Asix.Panel.Message(Replace(Asix.Script.Arguments(1),"$",""))
```

In JavaScript:

```
var args = new String(Asix.Script.Arguments(1))

var re = /\$/g
        Asix.Panel.Message(args.replace(re,""))
```

Though the AS program treats expressions $...$ as variable names, the real interpretation of parameters depends on the script author. For example, the following script allows to execute any modified operator action:

```
Asix.ExecuteAction(Replace(Asix.Script.Arguments(1),"$",""))
```

9.3. Composing Masks

In the phase of application execution, the operator has at his disposal a set of standard operations initiated with mouse or keyboard, which allow him to control operation of an application. The ability of opening and closing visual masks is here the most important point. However, standard mechanisms should be treated as supporting ones. Creating a full-logic application model requires application of mechanisms, which may be defined by user itself.

Designer can influence the way of using the application by providing the operator so called operator action. The set of all available actions is described in separate chapter. The actions may be defined as commands, which after their activation initiate the specified operations performed by AS32 program. Actions are defined by entering proper texts into editing fields provided.

On the stage of execution, action may be initiated by clicking the mouse on the proper place of the mask or pressing proper keyboard shortcut. The next part of the chapter describes keyboard usage.

9.3.1. Key Identification Rules

Execution of an action can be connected to pressing specified key combination in two ways. At the first, the action is used in object *BUTTON*. In addition a key combination is defined (in object definition window), which will initiate the action execution. The second way of using of the keyboard is defining so called keyboard shortcuts. The keyboard shortcuts allow to define unlimited list of different key combinations together with the appropriate operator actions. The keyboard shortcuts are joined to visual masks. There is a field in basic mask parameter window, where you should give the name of the shortcut in (a possibility of choosing from a list of shortcuts is also available, by pressing *Grey*-+ key or clicking the right mouse button).

Characteristic feature of using the keyboard shortcuts (in both versions) is, that chosen combinations are pending during the full time of mask presence (which they are joined to) on the screen. It doesn't matter, which mask is active.

The scheme of choosing action connected with key combination is as follows:

- if the system is not in executing mode, action recognition is disabled;
- if the window, which is not a visual mask, is active, the action recognition is disabled;
- for current (active) visual mask, checking is carried out whether the key has been used in the object, which belongs to the mask; if so, action defined in the object is executed and further check is stopped;
- for current (active) visual mask, checking is executed, if the key form shortcut joined to the masks; if so, action defined in the keyboard shortcut is executed and further check is stopped;
- the two previous points are repeated for all opened visual masks, until action associated with key combination is found; the sequence of mask reviewing is not specified (it strongly depends on history of masks switching-over); if mask of dialogue type is active, the verification of a key is not transferred to the remaining opened masks.

On the ground of the above scheme, the following conclusion can be drawn:

- browsing is executed until the first correlation of the key and an action is found; you cannot initiate executing two different actions, by clicking one key combination; if it required, you have to use compound action, where one executed action consists of lots of component actions;
- defining of action, which is to be available all the time, is performed by the simplest method, by defining required correlation of the key and the action on the mask, which is opened all the time;
- you can define, that the same key combination initiate different actions, depending on which mask is active at the moment.

9.3.2. Defining Keyboard Shortcuts

Keyboard shortcuts are stored in binary files. Name of shortcut is the same as file name. An extension of key file name is always *MNU*. All editing operations on keyboard shortcuts must be executed in designer mode of AS32 program. The basic window used for keyboard shortcuts management is the below presented selection window. It is opened by means of TOOLS. KEY SETS command from menu of designer window. Similar window is open directly from the field *Key sets* in defining window of visual masks (*Grey*-+ key, or clicking the right mouse button).

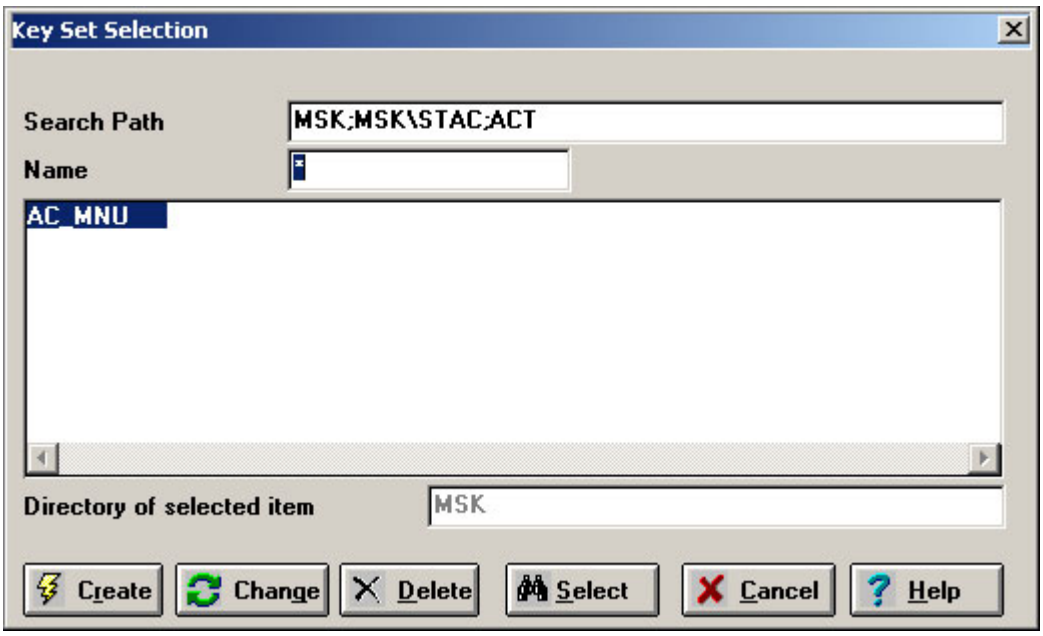


Figure 9-12. 'Key Set Selection' Window.

The essential part of the window is a list, which contains the names of all defined keyboard shortcuts available in the application. The shortcut is available, when its file is in one of the directories given in the *Search path* field. The path is defined as a list of names of the directories separated with semicolon mark. You have to remember that searching path is common for all application elements: visual masks, keyboard shortcuts and compound actions definitions. Changing the path in sets selection window, influences the access to the masks and actions. Performing operations on shortcuts consists in selecting one shortcut on the given list and pressing proper key located in the bottom part of the window. You can alternatively enter the name of the chosen shortcut manually in *Name* field. This field can also be used to reduce the quantity of the shortcuts on the list. By proper using of * and ? characters, you can reduce the shown shostcuts to chosen subset. *Directory of selected item* field is a supporting field, which shows, in which directory a file describing indicated shortcut on the list is stored.

Meaning of the keys in the selection window is as follows:

Delete - this operation cause deleting of indicated shortcut; the file describing the shortcut will be removed from the disk;

Change - entering this window allows to make changes in keyboard shortcut definitions;

Create - creating of new keyboard shortcuts; the file describing the shortcut is created in the directory, which is on the first position in the path.

Window used to create or change the keyboard shortcuts is shown below.

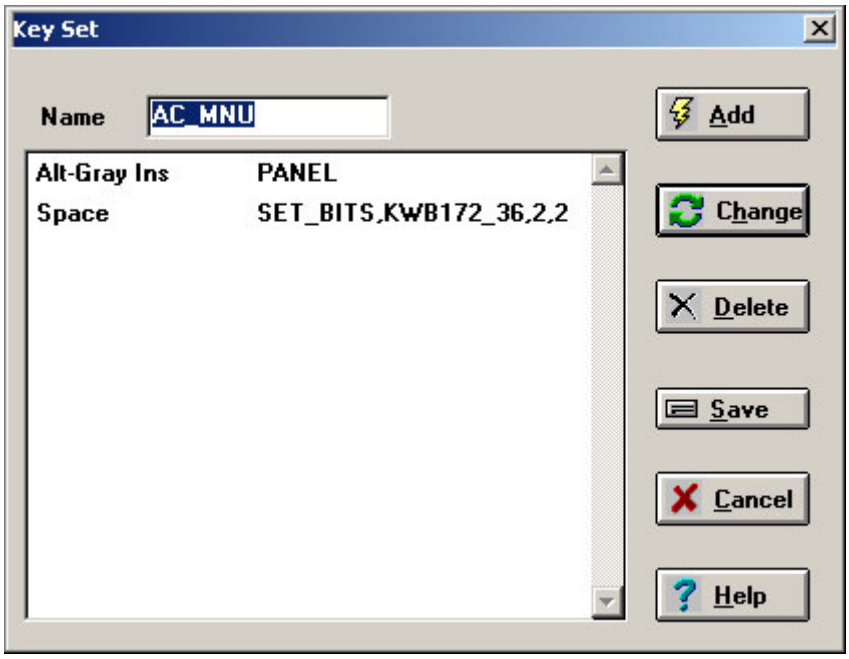


Figure 9-13. The Window for Key Set Definition.

In the *Name* field there is a name of defined keyboard shortcut, which is used to identify it within the framework of application and to create name of the file describing the shortcut. It is initially empty for new defined shortcut. By changing the name, it is possible to create a copy of previously defined shortcut. AS32 program is asking in this kind of situation, if program is to change the name of existing shotrcut or create a new one with given name and the same definition as original one. Successive editing operations relate to a shortcut with new name.

The alternate method of copying a keyboard shortcut is just copying the file, which include the shortcut preserving also the *MNU* extension.

The essential part of the window is a list, showing all defined key-action correlation in the shortcut. Each item in the list gives the text name of the key combination and the text of the associated action. Actions are sorted in alphabetical order of their names.

Keys on the right side of the window have the following meaning:

Change	- enters the window, which allows changing the key-action correlation definition for item indicated on the list;
Add	- creates a new item on the correlation list, and opens the correlation definition window;
Delete	- deletes the correlation indicated on the list;
Cancel	- closes the shortcut definition window with all entered changes abandoned;
Save	- closes the shortcut definition window and saves description of current shortcut in a file.

The window used to define the single key-action correlation is shown bellow. This window is opened with *Change* or *Add* key.

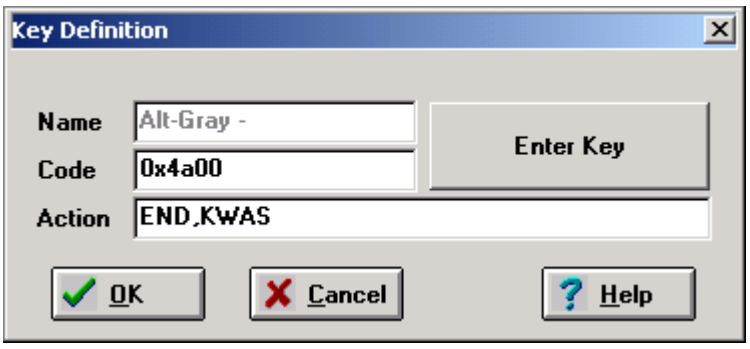


Figure 9-14. The Window for Assigning a Single Key to an Action.

The simplest way of defining the key combination, which will be combined with an action is performed by clicking on *Enter key* button. This will open the message window with a demand to press a chosen key combination. After pressing this combination the message window is closed and combination description appears in *Name* field in text version and *Code* field as hexadecimal number. Not all key combinations are allowed, for example: you cannot use *Alt-Fxx* combination.

In the *Action* field you have to enter text description of combined action. As in all cases of fields used to enter action, you can use name abbreviations and symbols. If the action description is correct, the text formatting will be performed taking into consideration full names after exiting *Action* field.

Using key combination, which consists in depressing a single letter key, needs an additional comment. The AS32 program treats capital and lowercase letters the same way. Defining correlation for „t" letter causes, that action will be performed for „t" and „T" letters. There is no possibility to define separate correlation for capital and small letters.

9.4. Control Operations

One of the functions of **asix** system operations is to control the process. It is carried out by sending proper control value to PLC. There are two methods of carrying out the control:

- sending values defined by the designer of the system;
- sending values determined as a result of a dialog with the operator.

First method consists in defining operator actions and providing them in keyboard shortcut or objects of BUTTON class. Three types of actions are available. In every case designer determines the destination address as well as the value to be send. Execution of action causes prompt transmission of given value. Action SEND_VALUE is used to transmit analog values. Actions SET_BITS and CHANGE_BITS are provided for digital control (*See: 9.4.1. Operator Actions*). They allow changing the selected bits of process variables (leaving the remaining bits unchanged).

The second method of control consists in placing on the mask the visual objects, which have assigned functions to execute control. When application is being executed the operator has the possibility to select an object and, with

dialog window, to set the control values. The dialog method depends on the class of the object used and it may be very diverse; for instance, selection of possible values of the set, entering numerical value. Selection of an object by the operator is performed by pointing it with a mouse or *Tab* and *Shift-Tab* keys (which cause sequential transition via all control objects). The moment in which the control is executed may be different.

Some classes of objects send their values immediately. The other ones save the selected value in memory and require an additional signal to be transmitted. This mechanism enables the operator to prepare a lot of controls and then to send them to the PLC simultaneously. Transmission of the signal is executed by means of operator action PERFORM_INPUT. Some classes of objects accept additionally an individual transmission signal – i.e. pressing the key *Grey*-+ (while an object is selected).

There is the possibility to lock all control operations of the application setting up the ***Control lock*** parameter in XML application configuration file:

Use Architect program > *Fields and Computers* > *Security* module > *Locks* tab

It is possible to configure the PERFORM_INPUT action in such a way that control may be performed from fully visible masks only. Controls will not be sent from the mask, which is even partially covered with another mask or any other window, although they have been set on it.

The second option is sending all controls set from all masks open at this moment.

The choice of the way of sending the controls is made in application configuration file, with use of ***No hidden controls***:

Use Architect program > *Fields and Computers* > *Masks* module > *Mics* tab

Sending controls also provides the mechanism, which enables setting the time of controls validity. This time is counted from the moment of execution of the last choice of a control by the operator. If the operator doesn't confirm sending the chosen controls, active controls are cancelled after this time. You can configure this option in:

Architect program > *Fields and Computers* > *Masks* module > *Mics* tab > ***Controls validity time*** parameter
If this parameter is missing or set with 0 parameter, the controls setting can remain valid for a period of time of any length.

All types of control operations may be protected with passwords.

9.4.1. Operator Actions

SET_BITS
SEND_VALUE
PERFORM_INPUT
CHANGE_BITS

See: 16.2. Description of Actions.

9.5. Passwords

The **asix** system is provided with module for password management designed to protect various parts of the system against unauthorized use. Two types of operations protected with passwords may be distinguished:

- operations controlling the process under control;
- operator actions (control of application execution).

In any case, during the attempt of performing the operation protected with a password, the following dialog window appears.

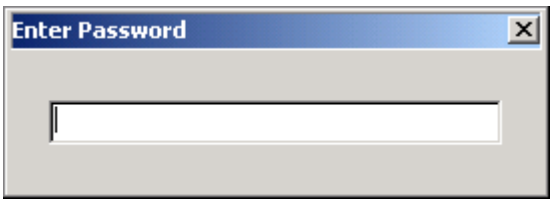


Figure 9-15. The Window for Entering the Password to the asix System.

The operator should enter the password and press *Enter* key. If the password is not entered correctly (wrong contents or password with different purpose), the operation, which is protected, will not be performed.

The password characters, which are entered by the operator are not displayed in dialog window (the position is only changed by the cursor) in order to prevent from the possibility of seeing it by unauthorized person.

Passwords are not checked in designer mode of operation.

The passwords are saved in the Passwords.ini file in coded version.

NOTICE In all operator actions where an individual password is declared it is possible to enter the number from the range 1-5 (instead of the password). It means the demand of entering a proper level password. The 5th level password means the access on an administrator level, and is accessible when using the logon system.

9.5.1. Protection of Control Operations

The control operations are protected on base of the set of 4 (maximum) hierarchical passwords. The system's designer declares a level of protection for each visual object, which can perform control operations. A number from 1 to 4 identifies the levels of protection. The level number 4 is the highest protection level. It is possible configure the object without protection of control operations. Passwords for individual level are declared in:

Architect program > *Fields and Computers* > *Security* module > *Level passwords mode* tab

When operator attempts to perform password protected control operation, the shown above password dialog window appears. The operator must enter the password appropriate for control to be performed. Password to be entered should be of level declared for object, which perform the control or any password of the higher level. If several controls are performed simultaneously (with different levels of protection), several password inquiries are possible.

If the password is entered correctly, the control operation will be performed. Moreover, every next control operation (on the same or lower level of protection) will be performed without asking for a password. This state will be kept till operator removes it or time limit expires, after which a password inquiry will take place again. The password validity time is declared in initialization file (by default – 15 minutes).

The operator has at his disposal the password management window, by means of which he can change password and cancel the validity time. The password management window is open with the *TOOLS.PASSWORD_MANAGER* command or *PASSWORDS* operator action. The password management window is shown below.

Figure 9-16. 'Password Manager' Window.

On the left side of the window, there is a column of check boxes showing current validity condition of passwords for individual levels. By changing these fields, you may remove or set validity condition of passwords. The attempt of setting validity field causes automatic password inquiry, after which in this case you have to give the password of the exact level, which the inquiry is concerning of. The keys *Change* are used to change the password. After

pressing the key, an inquiry for the current password is always performed (regardless of current validity condition). After proper reply (by means of the password for appropriate level) a password change window is open.

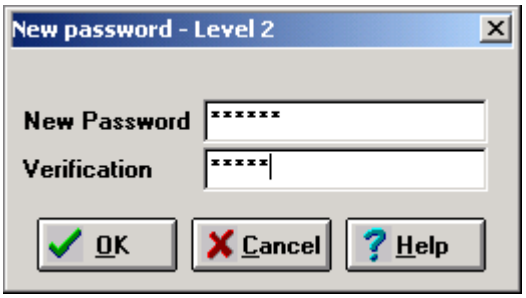


Figure 9-17. The Window for Password Change.

In fields *New password* and *Verification* you have to enter new password. Entering the password in two fields is carried out in order to prevent accidental mistakes during password typing.

You have to remember, that if the password is not set (is blank), so protection system on level of the password is not active. The system designer should define the initialization passwords during start-up.

9.5.2. Protection of Action Execution

A separate protection group is provided for operations that control the application execution. The problem is concerning to the locking execution of actions and some commands available in menu. This protection is configured in appropriate item in application initialization file and proper selection of actions.

The most general mechanism in this group of protection is action *GET_PASSWORD*. This action is provided for use in compound actions. By using the action *GET_PASSWORD* as the first element of compound action, you can lock performing all other elements, if the operator is not able to give the proper password. The content of proper password is one of the elements of definition of action *GET_PASSWORD (See: 9.5.3. Operator Actions)*.

The method of action *GET_PASSWORD* does not allow locking the commands in menu of Control Panel and Designer Window. Some of the key commands, namely: the switching over to designer and the ending the program, may be controlled by defining proper parameters in XML configuration file (***Designer lock*** and ***Exit lock*** – parameters are declared with use of Architect > *Fields and Computers* > *Security* module > *Locks* tab). The above-mentioned commands may be completely locked or protected with a password given in the contents of item. Completion of program operation can be also performed by means of action *EXIT.* In this action you may define independent password or use the password from configuration file.

9.5.3. Operator Actions

GET_PASSWORD
PASSWORDS
EXIT

See: 16.2. Description of Actions.

9.6. Logon System

The **asix** package allows developing the application equipped with the user logon system.

Activation of the logon system occurs by declaring the proper work mode of policies system with use of Architect program:

Architect > *Fields and Computers* > *Security* module > *Policies* tab > *Users mode* option

Users name and passwords are stored in the external Users.ini file.

The user database is managed by the administrative window opened by means of the TOOLS.USER_MANAGER command from the Designer window. The administrative window is accessible only after logging in the **asix** system as an administrator. First (after the LOGON_FILE entry has been declared), it demands creation of the user with administrator authorizations whom five levels of access are assigned to. To make it, the **asix** program should be started in Designer mode (without starting the application).

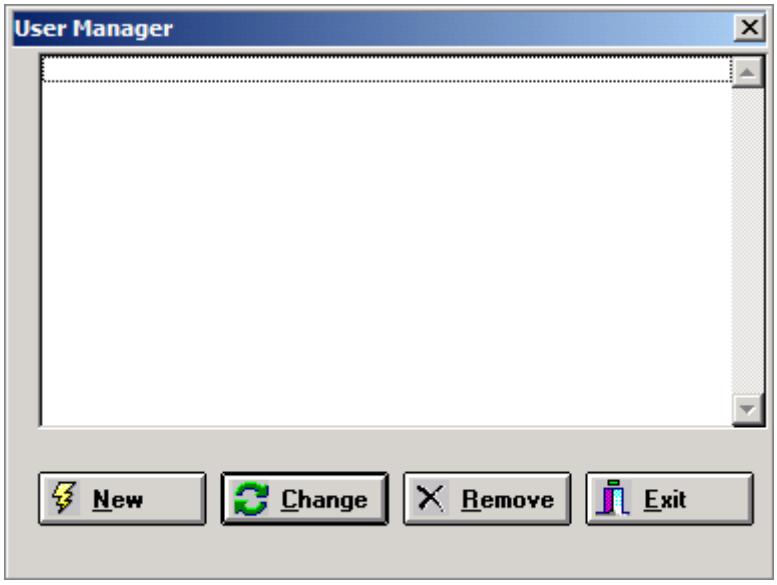


Figure 9-18. 'User Manager' Window.

The administrative window allows setting a new user, removing a user and changing user parameters.

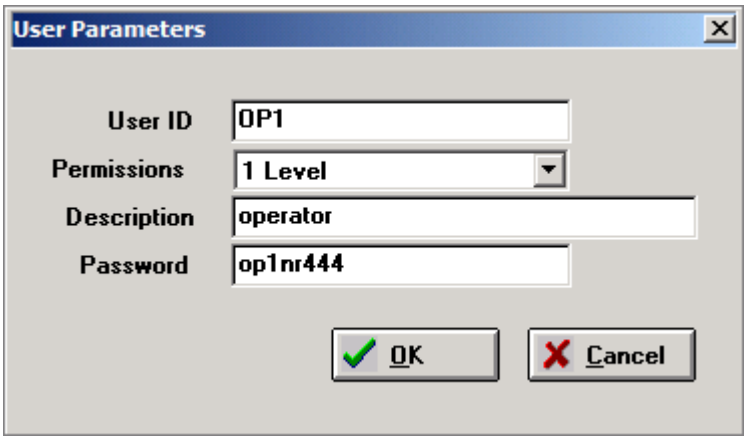


Figure 9-19. The Window for asix's Users Definition.

The user is described by the identifier (*User ID*) that consists of 6 characters, the description (of 30 characters), the password and the level of access (*Permissions*). There are 5 levels of access, from which first 4 correspond to the levels used in the password system and the 5th level – is assigned to the **asix** system administrator.

The *'User Logon'* window is opened by the TOOLS.USER_LOGON command form the Control Panel mode. The **asix** system also allows user to log out and change the password.

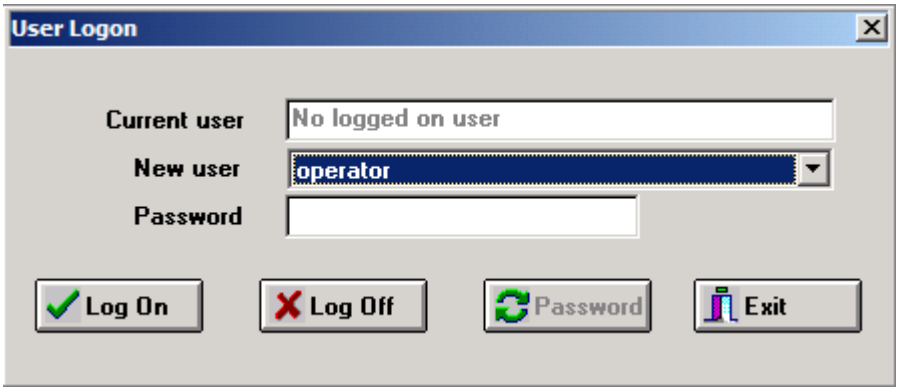


Figure 9-20. The 'User Logon' Window.

The *'User Logon'* window can be started by the LOGON operator action.

The operations of logon, logout and threefold incorrect entering the password are written down as system events into the alarm log file. The content of messages contains the user identifier.

The logon system assures the access to the data of currently logged user and his identifier. To enable it, it is necessary to define process variables:

- __USERID__ – string of 7 characters converted by the NOTHING_TEXT function,
- __USERNAME__ – string of 31 characters converted by the NOTHING_TEXT function.

The logon system is integrated with the password system (of level and individual passwords).

- The operation of the user logon of 1-4 levels corresponds to entering the password permanently. When the proper password is entered during logon activity - the operations protected by level passwords are executed without any other authorizations. The operations protected by individual passwords require entering the proper passwords each time.

- The user with the administrator authorizations of the operating system can execute all operations without necessity of entering a password.

- Working without logon makes all the operations protected with level passwords to be blocked. It is possible to execute operations protected with individual passwords by using the correct password.

- If the logon system is used, switching the **asix** system from the Executive mode to the Designer mode requires to be logged as the administrator of the **asix** system.

NOTICE In all operator actions, where an individual password is declared, it is possible to enter the number from the range 1-5 (instead of the password). It means the demand of entering a proper level password. The 5th level password means the access on an administrator level, and is accessible when using the logon system.

When the logon system is used, you can declare the default user that is automatically logged in when running a program.

The default user is declared by setting up the ***Default user id*** parameter in:

Architect program > *Fields and Computers* > *Security* module > *Users mode* tab

9.7. System Protections

Dynamic Protections
Static Protections

Mechanisms of system protections (policies) allow protection of access to potentially danger operation system functions, which used by an operator in incompetent and irresponsible way may lead to the system damage. Simultaneously, usage of protection mechanism allows system operation within user account with high authorizations. It makes maintenance and management of operation system easy.

Protection mechanisms consist of 2 groups:

dynamic – integrated with **asix**; they are switched on or switched off depending on the program state;

static – set one time and being active for all the operation system work time.

Dynamic Protections

Mechanism of dynamic protections is activated by the ***System policies*** parameter:

Architect program > *Fields and Computers* > *Security* module > *Policies* tab

Dynamic protections are switched on and switched off on-line. A set of such protections is as follows:

- Task Manager access lock (*Ctrl+Alt+Del* keyes);
- hiding and disabling Task Bar;
- disabling keys for the *Start* menu opening (*Ctrl+Esc* and *Windows*);
- disabling **asix**'s HLP files.

The way the protections are switched over depends on configuration way of access control system in the **asix**'s application. There are 2 possible variants:

- the logon system is active; dynamic protections are off only when an operator with administrator authorization is logged on;
- the level password system is active; dynamic protections are off only when a 4 level password is active.

The dynamic protection system is enabled after full application running, therefore protection of program start-up stage has been also designed. In the as32.exe batch line you should add the */PROTECT* switch – it causes activation of protections which can be unlocked only after administrator logon or entering a 4 level password.

The */PROTECT* switch may be used also when the dynamic protection system is off (POLICIES item is not declared). In such a case an operation system is protected permanently during an application running.

Static Protections

Static protections set by means of Static Policies Manager (StaticPolicies.exe).

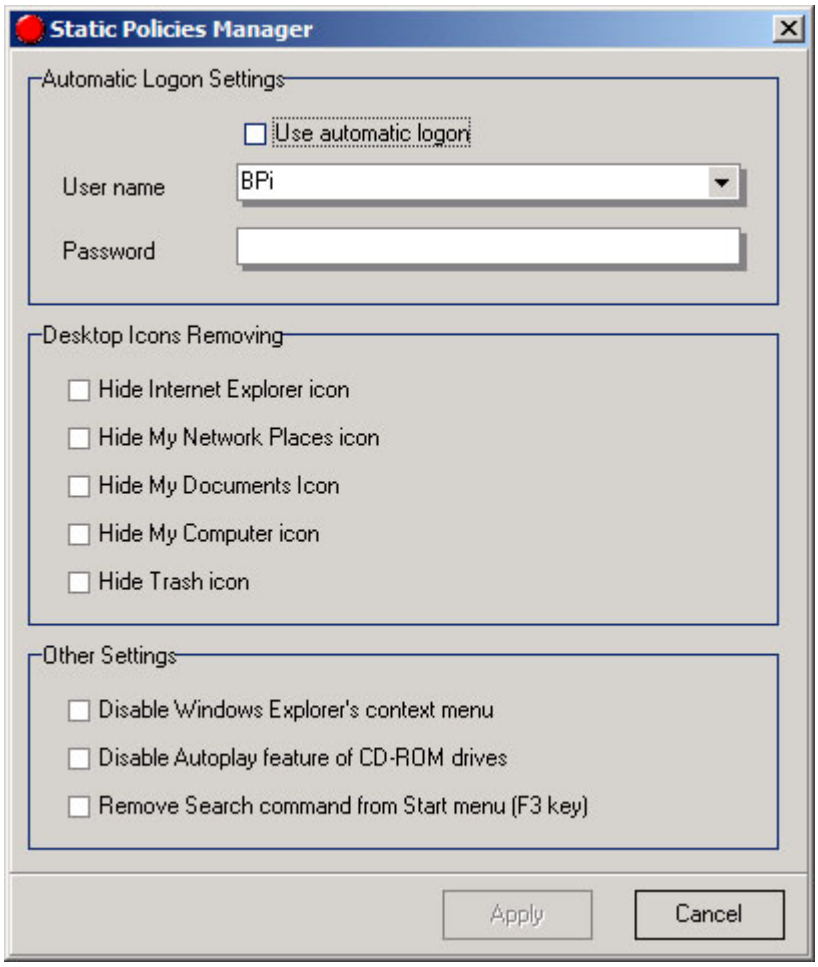


Figure 9-21. The Window of Global Policies Manager.

After selection of protections which are to be used, the *Apply* option should be confirmed, and then an operation system restarted. It is recommended to select all options.

Desktop Icons Removing and *Other Settings* options applies only to the Windows user whose account Static Policies Manager is executed within. A set of icons visible on the Desktop depends on operation system version and settings.

9.8. Using Bitmaps

Bitmaps are small pictures, used to present the graphical symbols, created by proper defining of rectangular bit matrix. The AS32 program uses the bitmaps for two different purposes:

icons **-** symbols, which are displayed instead of the window (visual mask) after minimizing operation is performed;

pictures **-** symbols, which are displayed by the objects on visual masks.

Bitmaps required during execution of application are stored in a picture library file or in a disc files of the following format: BMP, GIF, PNG, JPG (and others). A picture library file name is *asbitmap.dat* by default, but you may to declare other file name with use of ***Graphical symbols file*** parameter set up by Architect module:

Architect program > *Fields and Computers* > *Masks* module > *Fields and Directory* tab

There is a large library of bitmaps in the **asix** system package.

In the current version it is also possible to use graphic files with transparentness attribute as well as animated files. In particular, animated files can be used in the PICTURES object for animation of selected states.

External graphic files should always be put into directories declared in the ***Mask path*** parameter in the application configuration file:

Architect program > *Fields and Computers* > *Masks* module > *Fields and Directory* tab

(access path must not be entered in objects in the picture name field - external picture files are always searched in the directory of application masks).

The AS program is provided with a built-in bitmap editor, which can be started with TOOLS. BITMAPS EDITOR command (*See: 9.8.1. Bitmap Editor).* The detailed description of graphic editor is given in the separate chapter.

Using bitmaps of any size and any number of colors is possible. However, the built-in editor allows changing only 16 color bitmaps with size below 65000 pixels.

You can set the parameters of bitmap application by writing down the picture name in proper editing fields. For example, in mask definition windows there is a field *Icon*, provided to specify the icon name, which will to be used when operation of mask minimizing to icon is preformed. Similarly, in definition windows of visual objects, which displays information in a form of a picture, there is a field *Bitmap* (or similar), used to specify bitmap used by the object.

NOTICE Maximal allowed length of the picture name is 64 characters.

EXAMPLE

A typical example of window defining objects class *PICTURE* is shown below.

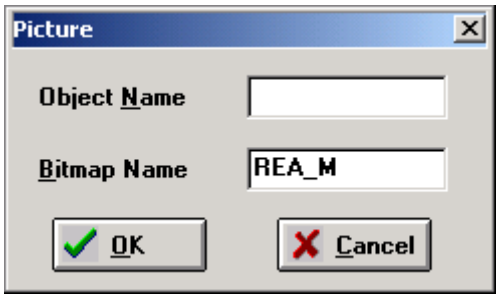


Figure 9-22. The Window for the PICTURE Class Object Definition.

In *Bitmap Name* edit field you should enter a bitmap name used by the object. In case of using bitmaps saved in separate files in BMP format a disk file name with extension should be specified. Bitmap files (*.BMP) has to be placed in one of the directories declared in MASK_PATH line in application's initialization file. You can also select the bitmap from list of bitmaps available in application by pressing *Grey +* key or clicking the right mouse button, after previous selecting the *Bitmap Name*. This opens a window including list of all bitmaps contained in application bitmap file. From this list you can select the bitmap by indication proper item.

The window of bitmap selection is shown below.

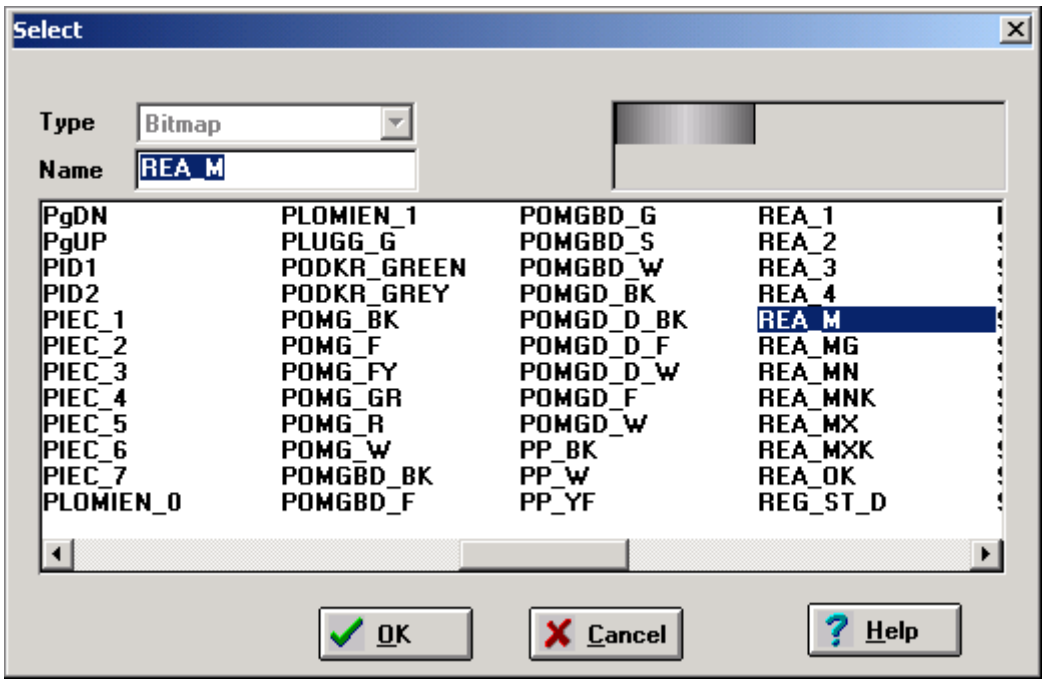


Figure 9-23. The Window for Bitmap Selection.

The drop-down list *Type* is used to choose bitmap type (icon or picture). If the type of bitmap results from the context of selection window use, so the map

type choice is disabled. In *Name* field you should enter the name of the map, which has been chosen. Alternatively, you can show a map on the displayed list of all bitmaps. During bitmap selection, in the window, which is in upper right window corner, there is a currently selected map content displayed.

The button set and the operation performed on selected map depends on the context, which the selection window was used in. The *OK* and *Delete* keys cause performing an operation of choice type, or map import (for more information on import in case of call the bitmap selection window by the MAP.IMPORT option of the editor, *see: 9.8.1. Bitmap Editor*). Double mouse click (on position of chosen name) is equal to pressing *OK* key. The *Cancel* and *Exit* keys cause window closing without performing any operation. In case of importing a map between library files (*see: 9.8.1. Bitmap Editor*), you may choose any amount of maps on the list of selection (space key is used to highlight the map choice by means of the keyboard).

9.8.1. Bitmap Editor

MAP
EDIT
OPTIONS
Keyboard Shortcuts Used for Map Editing

The AS32 program is provided with built-in bitmap editor. Maps may be minimized and used as icons, or as pictures displayed by visual objects. The editor is started with TOOLS.BITMAPS EDITOR command from Designer Window menu. All edit operations are performed on maps, which are stored in bitmap library used in designed application (library file is declared in an application configuration file with use of Architect: Architect program > *Fields and Computers* > *Masks* module > *Files and Directories* tab > ***Graphical symbols file*** parameter).

The window of bitmap editor is shown below:

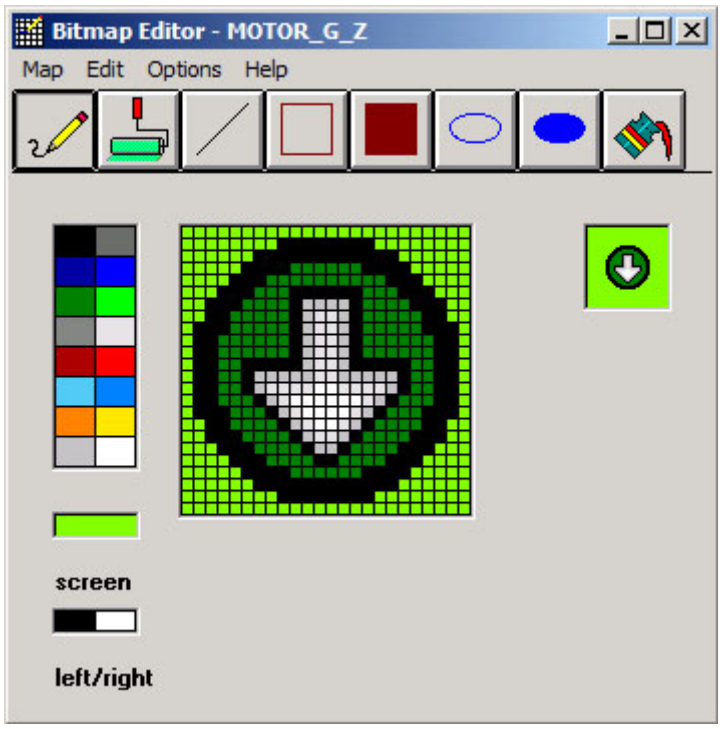


Figure 9-24. 'Bitmap Editor' Window.

The window include the following items:

title line - title line, in which the name of current edited bitmap is presented.

drop-down menu - the menu, which contains the commands used to edit bitmaps

toolbar - toolbar designed for drawing (drawing modes). The tool selection commands are also available in menu. The toolbar allows the fast changing the tool with the mouse.

edit area - an area used for direct bitmap editing. Its structure will be given in the further part of the chapter.

The meaning of the commands in the drop-down menu of the editor window is as follows:

MAP.NEW

Creating new map. This command, when executed, opens the following dialogue window:

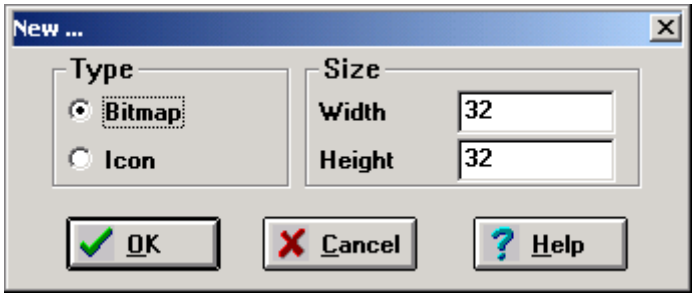


Figure 9-25. The Window for New Bitmap or Icon Creation.

When creating a new map, you have to specify its type and initial size. If the type *Icon* will be selected, the size is set to 32x32 pixels. It cannot be changed. The type *Bitmap* permits free declaring the size, which can be changed also later (with the command EDIT.CHANGE_SIZE).

MAP.OPEN
Beginning of edition of created previously bitmap. This command, when executed, opens the standard bitmap selection window. You should choose in it a bitmap, which will be edited. In selection window, you can switch between icon and pictures lists.

MAP.SAVE
Saving the changed bitmap in map library file.

MAP.SAVE_AS
Saving the edited bitmap in the bitmap library file with changed name. The previous version of bitmap with the original name remains in the library.

MAP.IMPORT
This command permits appending an external bitmap to the library file. The selection of file to be appended is performed in open file choice window. You may import the files of types BMP or ICO, or from another **asix** bitmap library (files of types DAT). The selection window allows selection of all pictures that do not exist in a current library (you should choose the option *none*, when being asked about overwriting existing pictures).

MAP.DELETE
Deleting the chosen bitmap from library file. The choice of the bitmap is performed in the opened selection window.

MAP.EXIT
Exit from bitmap editor.

EDIT.UNDO
Undoing the last editing operation performed on the bitmap.

EDIT.CLEAR
Clearing the bitmap to the state, in which it was after map creation (white rectangle).

EDIT.PENCIL
Choice of the tool (drawing mode) of pencil type.

EDYCJA.BRUSH
Choice of the tool of Brush type. The size of the Brush can be set with the command OPTIONS.BRUSH SIZE.

EDIT.LINE
Choice of the tool of *Line* type.

EDIT.RECTANGLE
Choice of the tool of *Rectangle* type.

EDIT.FILL RECTANGLE
Choice of the tool of *Filled Rectangle* type.

EDIT.ELLIPSE
Choice of the tool of *Ellipse* type.

EDIT.FILL ELLIPSE
Choice of the tool of *Filled Ellipse* type.

EDIT.FILL
Choice of the tool of *Fill* type.

EDIT.CHANGE_SIZE
The command, when executed, opens the dialogue window, in which a new size of currently edited bitmap can be given. The previous content of the bitmap is not destroyed, only proper amount of columns or lines are added (removed). The size of the icons cannot be changed.

EDIT.SET PALETTE
This command, when executed, opens the below dialog window, in which you can choose the colors of edited bitmap. The edited bitmap can have 16 different colors.

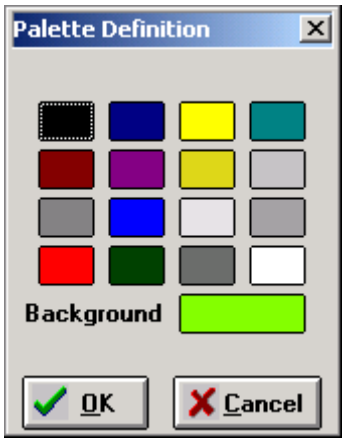


Figure 9-26. The Window for Edited Bitmap Color Definition.

OPTIONS.GRID
The command enables/disables displaying pixel grid in bitmap edition area.

OPTIONS.BRUSH_SIZE
Setting the size of the Brush tool.

The bitmap editor has been enriched with a context-sensitive menu that allows faster access to edition functions. It is very useful for operations: *add/delete row*, *add/delete column*. The context-sensitive menu can be opened by means of the right mouse button or the right mouse button in conjunction with the *Ctrl* button (the second variant should be used when the cursor is over one of elements of edition window).

The basic part of editor window is editing area. The following elements are its parts:

color choice rectangle - field located in left upper corner of editing area; it is built of multi-color fields used to choose the color while drawing; you can

choose a color by indicating appropriate rectangle and clicking with left or right mouse button; both keys have independently assigned colors;

„screen" background choice field - clicking with one of the mouse buttons in field area causes, that to this key the attribute *Transparent* is attached;

„left/right" color choice - status field showing currently assigned colors for both mouse keys;

map edit field - main editor field used for edition of bitmap; it shows in enlargement, the current state of map; the degree of enlargement depends on the size of the bitmap and the size of editor window;

map monitoring field - the field is located in upper right corner of editing area; it shows the current appearance of the edited bitmap in its real size; the background area surrounds the bitmap.

Bitmap edition is limited to mouse controlled operations in edit area. The essential meaning has the currently selected tool (drawing mode). It influences the way of mouse operation. All edit operations can be performed by means of the left or right mouse button. Each button has a certain attribute assigned to (set in color and background choice field). Key choice determines the attribute, which the operation will be performed with. The meaning of all types of tools and the specific for them way of operating with mouse are given below:

Pencil - mouse clicking causes setting of the color of single pixel pointed with mouse;

Brush - mouse clicking causes setting of the color of pixels in the closest surroundings of the mouse position; the size of the brush is defined by means of the OPTIONS.BRUSH SIZE command;

Line - tool allowing changing the color of all pixels, which belongs to line segment; the starting point is in the place of mouse clicking, the ending point is in the place of key releasing;

Rectangle - tool allowing to draw a rectangle in chosen color; two opposite corners are specified by the places of pressing and releasing mouse buttons;

Fill rectangle - the tool is managed the same way as the previous one; the difference is, that the interior is also filled with the color attached to the button;

Ellipse - the tool, which allows drawing an ellipse in a chosen color; two opposite rectangle corners describing the ellipse are specified by the places of pressing and releasing the mouse buttons;

Fill ellipse - the tool is managed the same way as the previous one; the difference is in the fact, that the interior is also filled with the color attached to the button;

Fill - tool that allows changing the color of the chosen bitmap area; clicking mouse button in chosen bitmap place indicates the area; the color is changed for all points, which color is the same as the color of indicated point and additionally the point is not cut off (separated) from the indicated point with points in different color.

The transparency attribute needs additional explanation. Its usage in any point of the bitmap causes, that in the moment of displaying, in transparent points the current background will be shown. In monitoring field, these points are displayed in color used in background choice field. In case of necessity, the background color can be changed in palette set window.

Keyboard Shortcuts Used for Map Editing

Bitmap editor has conversion function of edited bitmap. These functions are accessible with the bellow keyboard shortcut:

Alt_R	- bitmap rotation of 90 degrees;
Alt_Z	- mirror image along horizontal axis;
Alt_X	- mirror image along vertical axis;
Alt_W	- record inserting; if a mouse cursor is outside the area of the edited bitmap, then a record in inserted at the top of the bitmap; if the mouse cursor is inside the area of the edited bitmap, the record is inserted below the mouse cursor;
Alt_S	- record deleting; if a mouse cursor is outside the area of the edited bitmap, then a record at the top of the bitmap is deleted; if the mouse cursor is inside the area of the edited bitmap, the record below the mouse cursor is deleted;
Alt_C	- column inserting; if a mouse cursor is outside the area of the edited bitmap, then a column in inserted at the left-side of the bitmap; if the mouse cursor is inside the area of the edited bitmap, the column is inserted at the right-side of the mouse cursor;
Alt_V	- column deleting; if a mouse cursor is outside the area of the edited bitmap, then a column at the left-side of the bitmap is deleted; if the mouse cursor is inside the area of the edited bitmap, the column at the right-side of the mouse cursor is deleted.

9.9. Using Expressions

Clicking with the right mouse button in edit field, into which the ASTEL language expression sequence is entered, you open the window, which allows generating these expressions. This relates to the proper fields of object type EXPRESSION, CALCULATOR and REPORT and text editor, working especially as a report editor.

Using the window, which is displayed, after choosing the command, the user can:
- create and calculate the ASTEL language expressions,
- check the reasons of errors occurred during evaluation,
- correct the previously entered expressions,
- put the created expression to the edit field.

The dialogue window is divided into several areas. There are fields that allow generating the expressions grouped in the upper part of the window. Selected

items are added to the expression field. User acquainted with the ASTEL language can write the expression directly in this field.

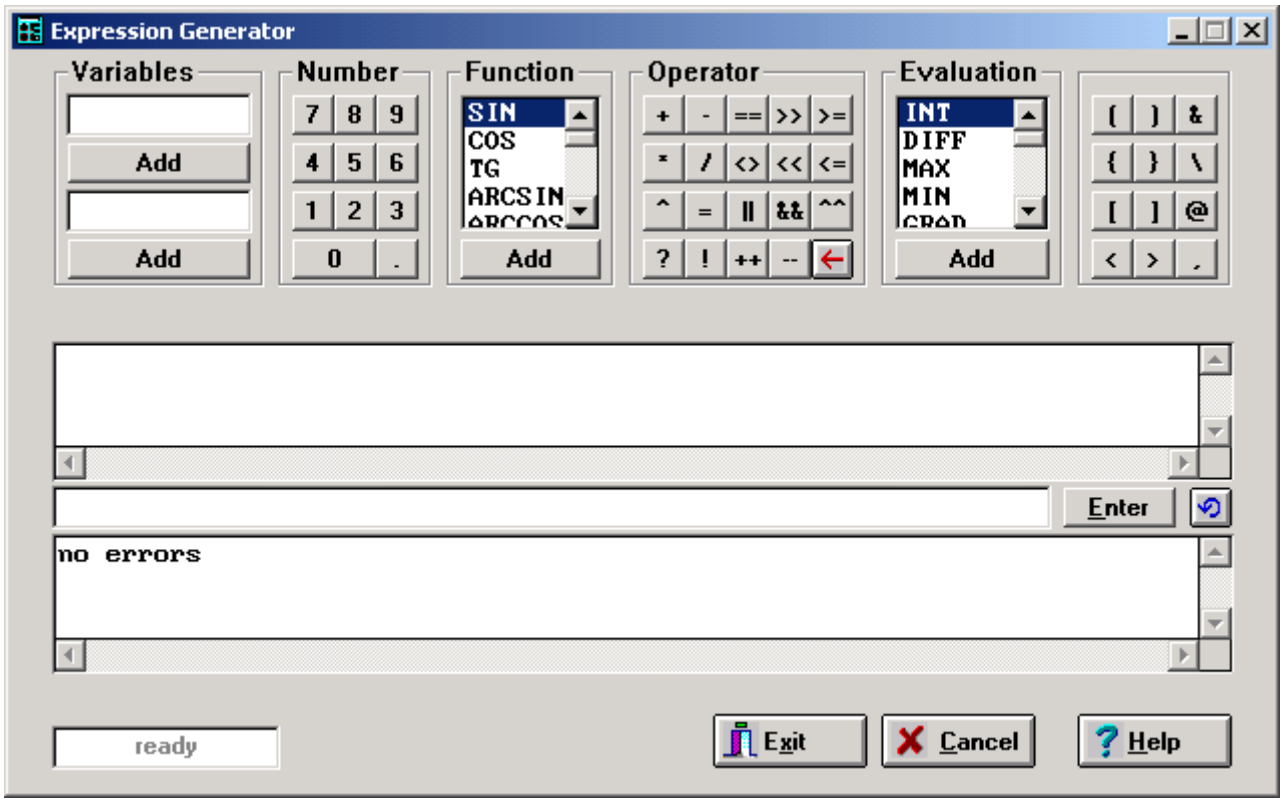


Figure 9-27. The Expression Generator Window.

Upper part of the window, has the fields discussed in turn (from the left):

- the field Variables allows adding variables written in this field to an expression created in the line; the variables can be written manually in two fields provided for this purpose, and added by clicking the Add button placed under them; it is also possible, to select the variable in variable selection window, which appears after clicking the right mouse button in variable name field; clicking in upper field causes displaying ASMEN variable list, clicking in bottom field – ASPAD variable list;
- field that allows adding numbers, composed of keys adding digits from „0" to „9" and the „. " letter, used in decimal numbers;
- the field Function allows to add the function name for example SIN; to write down the name of the function, you have to choose it from the list and then press the button Add placed under it, in order to add the name of the function;
- field that allows adding operators in expression; the key marked with red arrow "←" is equal to pressing BACKSPACE key in expression field and causes deleting the last letter;
- the field Evaluation allows adding the evaluator name, for example MAX; choose the name of the evaluator from the list and then press the button Add placed under it, in order to add the name of the evaluator;
- the field that allows adding other various characters to the expression, which may occur in it, for example, brackets or elements of so called moments; adding is performed after pressing the proper key;
- expression evaluating is performed by pressing ENTER key located on the right side of expression field; after the calculations are completed, the expression

field remains empty; editing key located nearby marked with a blue arrow with rotation causes adding recently evaluated expression in expression field; it can be used as the source of text for edition;

- above the expression field, there is a text field saving the course of successive evaluations; in next lines (from the top) the evaluated expressions and evaluation result are added; if during evaluation a mistake occurs, then instead of a result, the „???" sequence is displayed;
  contents of this window is as follows:
  " the expression that has been evaluated lately";
  "result of the latest evaluation";
  " the expression that had been evaluated before";
  "result of the previous evaluation";
  „…";
- below the expression field, there is a text field, which displays the list of errors and warnings, which has occurred during last evaluation or the no errors message;
- the edition button, marked with a blue round arrow, causes that the last expression that has been evaluated lately is inserted into the expression box. It can serve as source of text for edition.

Other fields assure:

- ability to set period for evaluators Period.from, Period.to;
- displaying the window operation status of type ready, computing and disabled;
- entering time range.

9.10. Using Variables

Process variables are declared in a database.

See: Architect user's manual, chapter *2. Handling of the asix application variable definitions databases*.

In every place, where specifying variable's name is necessary (at configuring dynamic objects) it is possible to use the window of variables list, which enables browsing, searching and selecting the variable of required name (TOOLS.VARIABLES).

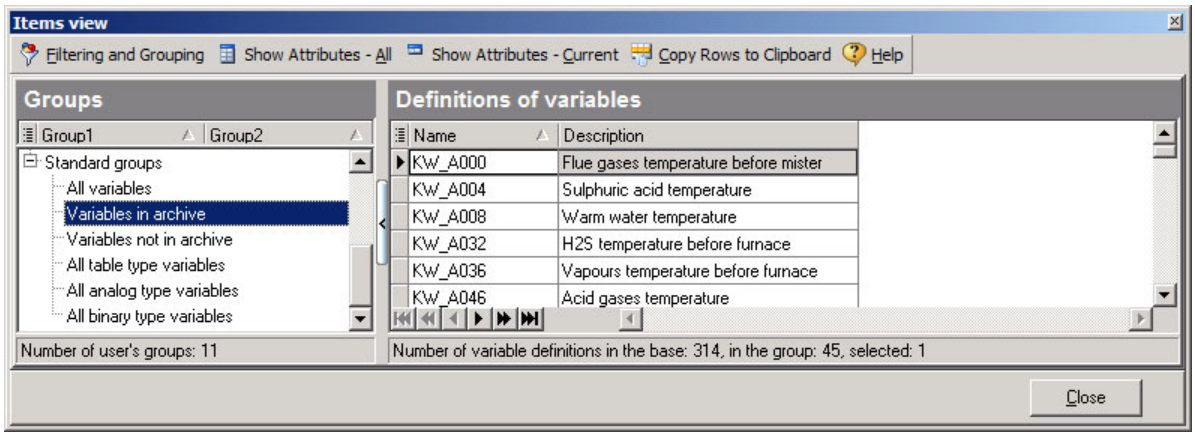


Figure 9-28. 'Items view' Window.

The main part of 'Items view' window is a list of variables declared in VarDef database. The variables may be sorted by any field – you should only click on the name of attribute (column caption) you would like to sort variable rows by. The key of sorting may be variable name or description, for example. The direction of sorting depends on markers: and .

The items view window includes a mechanism for incremental searching of custom attribute value. If the first letters of the searched value are entered into any column cell, the mechanism will automatically move to the first record with the given attribute value. This mechanism works for all attribute columns.

Searching and filtering records according to the required attribute value is performed with use of Filter builder launched by means of button.

9.11. Fonts Usage

Changing Font Definition
Changing Font Name
Adding Font to the List
Removing Font from the List

Text information in the **asix** system may be displayed with use of various typefaces.

Windows defining majority of visual objects, displaying information in alphanumeric form, are provided with field named *Font* (or similar one), used to specify the type of the font used in the object. A typical example of a window defining the object class *TEXT* is shown below.

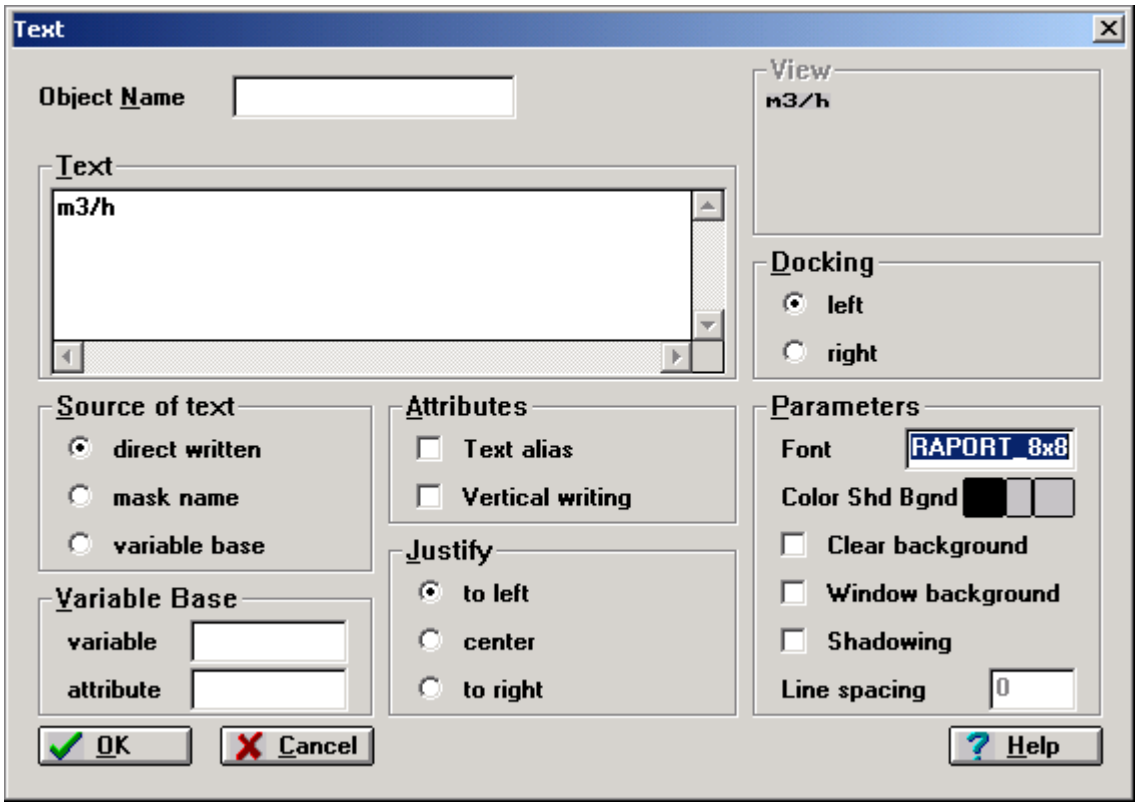


Figure 9-29. The Window for the TEXT Class Object Parameterization.

In editing field Font you have to enter the name of the font used by the object. The name may be as follows:

- the name of predefined font of variable letter width: SYSTEM, DIALOG or SMALL;
- the name of predefined font of constant letter width: ROM8x8, ROM8x14 or ROM8x16;
- the name of predefined font of constant width and code page 852: REPORT_8x8, REPORT_8x14 or REPORT_8x16, provided for reports and including frame signs;
- symbolic name of Windows system font; there is a set of additional raster fonts in asix packet; you may also use any other font.

Installed fonts may also be selected. Pressing *Grey-+* key or clicking the right mouse button, after previous font field selection carries it out. This causes opening the window showing list of installed fonts, which you can make a choice from. This window also allows performance of other management operations related to font usage. Font management window is shown below:

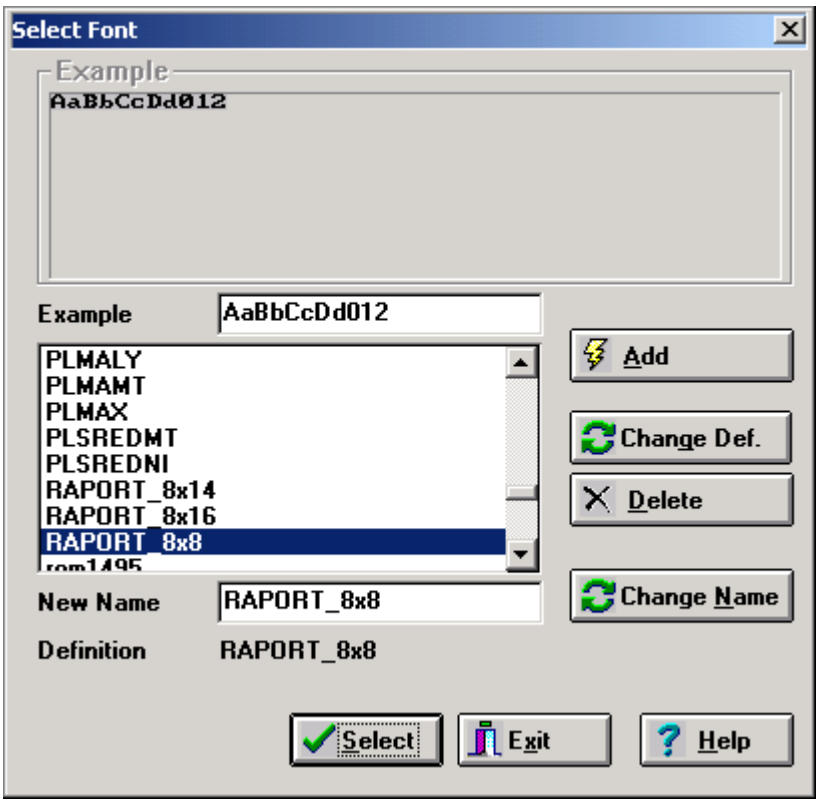


Figure 9-30. The Window for Font Handling.

After the *Select font* window appears you should make the selection of one of the fonts name from the list and then press *Select* key.

Choosing *Exit* button or pressing *Esc* key will cause closing the window without choosing new font.

In *Example* field you can see the chosen font. Letters displayed in *Example* window can be changed in editing field *Example*.

Changing Font Definition

After appearing the *'Select font'* window, you should select one of the fonts names from the list, and then click on *Change definition* button. The system font window will appear. After selection, the new definition will be assigned to previously chosen name.

Changing Font Name

After displaying the *Select font* window you should select one of the fonts names from the list. Its name will appear in *New name* edit field. You can edit this name and then click on *Change name* button in order to confirm new name in the *Logical name change* window, which will appear in the centre of the screen. The font name will then be changed on the list, but its definition will remain unchanged.

Adding Font to the List

After displaying the *Select font* window you should select one of the fonts names from the list and then choose *Add*. The window, similar to the font definition changing will appear, but the initial choice will correspond to the parameters of previously chosen font. After the new choice and its confirmation is made the new font will appear on the list with its name given by **asix** system. This name can be changed by means of above described method.

Removing Font from the List

After displaying the *Select font* window you should select one of the fonts names from the list, and then choose *Delete* and confirm choice. Then you can remove the name from the list in *Deleting fonts from list* window, which will appear in the centre of the window.

The symbolic name of the font is any name, which, in initialization file, the font description has been attributed to. During designing the application, you can use such defined names by font choice. The advantages of using this choice method are:

- using names which in logical way specifies the purpose and the shape of the font, for example: TITLE, FANCY, LOWCASE, MEASUREMENT; direct fonts descriptions are hard to remember and have no relationship with the way of their application;
- changing the font used for the certain purpose consists in changing the symbolic name definition in initialization file. All objects using this name will start automatically be displayed by means of the new font.

9.11.1. Directions for Polish Character Use

Types in 1250 code page are used to display polish letters in Windows XP/2000/NT4.0. **asix** system uses 852 code page, but if it is needed and possible, the automatic text conversion into 1250 code page is performed.

Report system uses types in LATIN II standard (852 code page) for all the time to ensure the compatibility with DOS version. These reports are converted into 1250 code page before displaying

Therefore, the version for Windows XP/2000/NT4.0 has additionally built-in types with constant character width, provided with frames named RAPORT_8x8, RAPORT_8x14 and RAPORT_8x16 - that use 852 code page. These types should not be used in objects, if the polish characters are to be proper displayed. Built-in text editor of **asix** system properly displays polish characters in the1250 or 852 code page, that is dependent on editor startup – from menu TOOLS or report window. The window that displays the content of text file, used in **DISPLAY_FILE** action, displays the polish characters in the 1250 code page. The same widow uses 852 code page when displaying the reports.

In case of printing in text mode e.g. alarm printing, report printing – a printer neeed dto have built-in types in accordance of LATIN II standard.

9.12. The List of Key Operations

Operations Concerning the Edit Fields
Operations Concerning Copying Text in Edit Fields
Operations Concerning the List with Possibility of Moving and Removing Items
Operations Concerning Object Editing in Designer Mode
Global System Operations
Operations Related to Key Macros

Most of the operations performed by the operator in **asix** system can be performed in two ways: by means of the mouse or using the keyboard. In this chapter the meaning of some specific key combinations will be given, which meaning had been already defined in the system. You have to bear in mind, that besides these combinations, the designer of the application system can define his own additional keys, which will initiate performing of actions chosen by him/her.

The interpretation of the keys is performed in **asix** system by means of context method. It means, that the same key combination can denote different operations depending on current system conditions. Similarly, in some situations, some of the keys can loose their meaning.

Here is the list of special **asix** system keys given below.

Enter	- confirm or execute operation (depending on usage context);
Esc	- abandon operation;
space	- selecting item from the list or perform operation in buttons;
Grey-+	- signal of selection, opening the choice list, confirming value transmission for visual objects;
arrows	- position or selection change in given direction;
Tab	- passing to next object/field;
Shift-Tab	- passing to previous object/field;
Home	- transition to the beginning of the list or the beginning of the line in edited fields;
End	- passing to the end of the list or the end of the line in edited fields;
Ctrl-Home	- passing to the beginning of text or line or transition to the beginning of the list;
Ctrl-End	- passing to the end of the text or line or transition to the end of the list;
Pg Up	- selection or cursor position change one page up;
Pg Dn	- selection or cursor position change one page down;
Alt-letter	- instant field selection in windows or performing the command from the menu. *Letter* used in combination is a distinctive letter of field name or menu command.

Operations Concerning Edit Fields:

Del	- deleting a letter on the position of the cursor;
Backspace	- deleting a letter before of the cursor;
Ctrl-Del	- deleting the text to the end of the line;
Ctrl-←	- passing to a word on the left side of the cursor;
Ctrl-→	- passing to a word on the right side of the cursor;
Ins	- changing the insert/write-over mode;

Ctrl-PgUp - passing to the beginning of the text;
Ctrl-PgDn - passing to the end of the text.

Operations Concerning Text Copying in Edit Fields:

Shift-arrow - cursor position change, connected with text selecting;
Ctrl-x - deleting with saving selected text into the buffer;
Ctrl-c - saving selected text into the buffer;
Ctrl-v - inserting saved text into the buffer;

Operations Concerning the Lists with the Possibility of Moving and Removing Elements:

Ctrl-Home - moving the element to the top of the list;
Ctrl-End - moving the element to the end of the list;
Ctrl-↑ - moving the list element up;
Ctrl-↓ - moving the list element down;
Alt-a - moving the list element up;
Alt-z - moving the list element down;
Ctrl-Del - removing the list element;
Alt-d - removing the list element;

Operations Concerning Object Editing in Designer Mode:

Ctrl-p - creating the object copy;
Ctrl-r - creating the object copy;
Ctrl-t - deleting with saving the object into the buffer;
Ctrl-x - deleting with saving the object into the buffer;
Shift-Del - deleting with saving the object into the buffer;
Ctrl_k - saving object into the buffer;
Ctrl_c - saving object into the buffer;
Shift-Ins - saving object into the buffer;
Ctrl_w - inserting saved object from the buffer;
Ctrl_v - inserting saved object from the buffer;
Ctrl-Ins - inserting saved object from the buffer;
Ctrl_u - deleting object;
Ctrl_q - deleting object;
Del - deleting object;
Ctrl-Home - moving the object to the top;
Ctrl_End - moving the object to the bottom.

Global System Operations:

Alt-F1 - opening the help window;
Alt-F2 - saving the current mask in a file (in designer mode);
Alt-F3 - refreshing the display;
Alt-F4 - closing the window;
Alt-F5 - printing the screen.

Operations Related to Key Macros:

Alt-F12 - opening the macro management window or ending recording sequence;
Alt-0... Alt-9- - macro replay;

Apart from above described keys, you can use in the system many other specific sequences, which have the importance in a very small range. This concerns mainly the operations performed by the visual objects (in particular, objects of *CHART* class). These keys will be described in chapters concerning these parts of **asix** system, which they are used in.

9.12.1. Macros

The AS32 program is provided with built-in mechanism of macro operations, which allows accelerating considerably the work of designer in case of necessity of multiple repeating the same sequence of operations. The system of macros is provided with the following features:

- operations performed with keyboard and mouse during macro defining are saved,
- macro is executed with the maximum speed (without saving original time dependencies),
- you can save maximum 10 macro operations,
- macros are available only in designer mode.

The process of macro saving is initialized by pressing *Alt-F12* key. This causes displaying the following window:

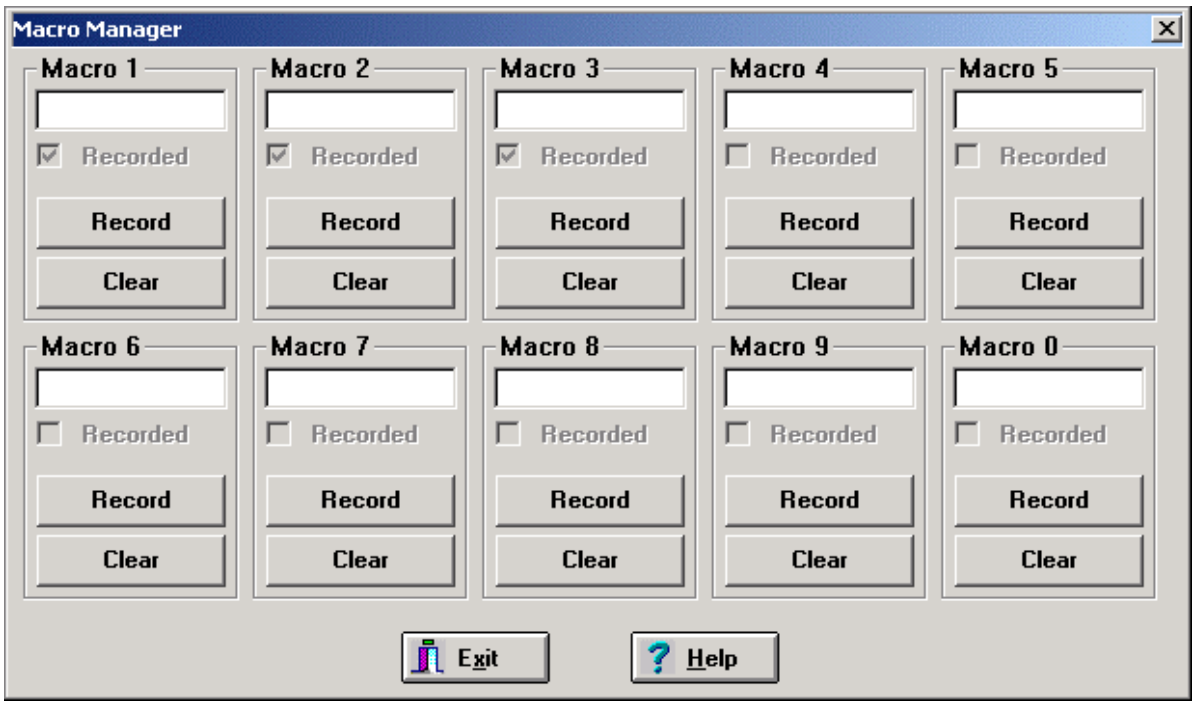


Figure 9-31. 'Macro Manager' Window.

The window is divided into ten sections corresponding to the individual macros. Fields, which are the part of every section, have the following meaning:

name field - macro name, used only as a supporting field, in which the designer can write what operation the macro performs,

check box *Recorded*	- indicator, which tells you if the macro is already saved,
the *Record* button	- the key, which initiate macro recording,
the *Clear* button	- the key deletes the previously saved macro operation.

After choosing the macro number, the designer should click on the proper *Record* button. The macros management window is closed then the designer performs the operations, which are to be saved. During saving, the AS32 program indicates with sound, saving each operation performed with keyboard or mouse. If the designer ends performing sequence of operations, which was to be saved, he should press *Alt-F12* key again. This will be indicated with special sound then the recording process will be ended. From this moment the designer has the ability to play the recorded macro. Executing the macro operation is initiated by pressing one of the key combinations: from *Alt-0* to *Alt-9*. The number of the key in used combination determines the macro number, which is to be performed.

The macros are stored in disk files.

COMMENTS *In the process of macro recording, all delays between the operations are removed. This causes sometimes problems, if during macro execution some system windows are to be opened (for example: color choice window). They aren't able to manage so fast with events. In that type of situations, you can use macros numbers 0 and 9. During macro execution they are adding small time delays between next operations.*

10. Executive - Running the Application

AS32 Program starts automatically in run-time mode, if it is started with initialization filename specified in start up command. The basic features of run-time mode are as follows:

- all synoptic masks given in section AUTO_START of initialization file are automatically opened,
- the synoptic masks are opened in updating (refreshing) mode,
- operator's actions are being executed,
- alarm system is active,
- communication and archiving modules are started (ASLINK, ASMEN, ASPAD).

10.1. Control Panel

In the moment of start of AS32 program in application run-time mode a window of Control Panel is displayed. This is the main window of application, which fulfils the functions as follows:

- provides a set of commands used to control operation of application. These commands make an supplement of operation set provided by system designer in form of operators actions.
- displays and stores the messages related to internal system operation.

Control panel remains opened during the whole period of application operation (it may be temporary closed after passing to designer's mode, but then it will be re-opened after come back to application mode). Closing Control Panel is equivalent to the end of application operation.

Selection of Control Panel by the operator is performed by means of standard mechanisms of AS32 program environment (for instance, pointing with a mouse, pressing *Alt-Tab* key), or by means of execution of the PANEL operator action provided by a designer.

Location and size of Control Panel window at the moment of application start may be declared in the configuration file in ***Control panel window position*** parameter with use of Architect program:

Architect > *Fields and Computers* > *Start parameters* module > *Control panel* tab

There is also an opportunity to automatic minimizing of Control Panel window.

REMARK An attempt of AS32 restarting, while the previous instance is in *hide* mode, makes the program active from the *hide* mode. It is very useful when the Windows taskbar is disabled.

10.2. Components of the Control Panel

Control Panel window has been presented below:

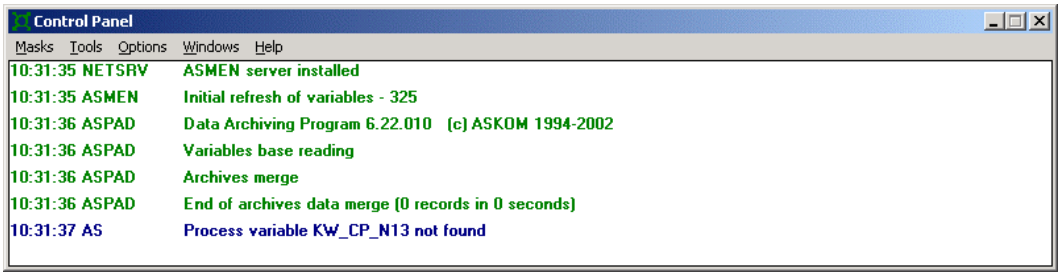


Figure 10-1. 'Control Panel' Window.

The basic component of the Control Panel is the list of system messages, which relates to events connected with program operation as well as the state of system environment. These messages occur during application operation. Typical examples of messages are: installation messages determining active system components, information about errors occurred during process data transmission, messages concerning system auto-diagnostics. Each message is composed of time stamp, identifier of program component, the part of which the event relates, as well as the message text.

Messages are divided into types, which correspond with their level of importance. The messages are displayed in various colors depending on type:

green color - message;
blue color - error;
red color - critical error.

Occurrence of event of critical level is signaled additionally with sound signal.

Number of recent messages to be saved in the Control Panel list may be set in application configuration file in ***Maximal size*** parameter with use of Architect program:

Architect > *Fields and Computers* > *Start parameters* module > *Control panel* tab

10.2.1. Description of the Control Panel

MASKS.OPEN
MASKS.DESIGNER
MASKS.EXIT
TOOLS.ACTIVE_ALARMS_WINDOW
TOOLS.PRINTER_MANAGER
TOOLS.PASSWORDS_MANAGER
TOOLS.USER_LOGON
TOOLS.DIALOGS_WINDOW
TOOLS.HISTORICAL_ALARMS_WINDOW
TOOLS.REPORTS_WINDOW
TOOLS.FILE_MANAGER
TOOLS.VARIABLES
TOOLS.SYSTEM_STATE.ALARMS
OPTIONS.TIME_CHANGE
OPTIONS.DEMO

Below all commands included in the menu of the Control Panel has been described. More detail information concerning to those commands may be found in corresponding topic sections.

MASKS.OPEN
Opens any selected synoptic mask. As result of execution of this command, a window is opened showing names of all defined masks, which operator may select. Selection window is described in section related to Designer Window. The preferred methods of opening of synoptic masks in an application are, operators' actions however.

MASKS.DESIGNER
Transition to designer's mode. It enables editing the masks in parallel with normal operation of the system. This command may be locked or protected with a password by means of appropriate definition of ***Designer lock*** parameter in application configuration file declared with use of Architect program:

Architect > *Fields and Computers* > *Security* module > *Locks* tab

MASKS.EXIT
Close the Control Panel window that denotes the end of application operation. This command may be locked or protected with a password by means of proper definition of ***Exit lock*** parameter in application configuration file, with use of Architect program:

Architect > *Fields and Computers* > *Security* module > *Locks* tab

EXIT operator's action is the equivalent of the described command.

TOOLS.ACTIVE_ALARMS_WINDOW
Opens the active alarm window in its standard complete form. This is an alternative method of active alarm handling, beside the main method of freely defined windows.

TOOLS. HISTORICAL_ALARMS_WINDOW
Opens the window of historical alarms in its standard form. This is an alternative method of historic alarm handling, besides the main method of freely defined windows.

TOOLS.REPORTS_WINDOW
Opens the report system window, which permits calculation of selected reports as well as other operations, related to reports. This command may be locked by means of Change lock parameter in application configuration file with use of Architect program:

Architect program > *Fields and Computers* > *Security* module > *Locks* tab

TOOLS.DIALOGS_WINDOW
Opens the Dialog window, which permits interactive calculation with use of variable values.

TOOLS.VARIABLES
Opens the window, which permits browsing of process variables list (the current and the archive ones) defined in the system.

TOOLS.PRINTER_MANAGER
Opens the window of printer manager. This window enables browsing of current printouts, management of queue of waiting printouts as well as printing of files.

TOOLS.PASSWORDS_MANAGER
Opens the window of password management. This window permits setting the condition of password priorities as well as the changing the passwords.

TOOLS.USER_LOGON
When the logon system is used, the command allows startup window *'User logon'*.

TOOLS.SYSTEM_STATE.ALARMS
Opens the window in which the current alarms in system operating mode are shown as well as its configuration.

OPTIONS.TIME_CHANGE
Opens the window, which permits to set the current time.

OPTIONS.DEMO
Records or runs the demo sequences.

10.2.2. Switching Over the Visual Masks

The set of visual masks makes the basic part of each application. In run-time mode operator selects the masks, which are to be displayed. In the Control Panel window the MASKS.OPEN command is provided to permit selection of any mask included in application (file describing the mask is located in one of directories indicated in the MASK_PATH item of the START section). This is however an auxiliary command. The basic mechanisms of switching over the masks are operators' actions OPEN_MASK and CLOSE_MASK. Designer of the

system has to make these actions available by means of defined key sets as well as objects of class *BUTTON.* Designer must also determine (in section AUTO_START), which masks are to be opened automatically at the moment of system starting.

Usually, the masks are loaded from disk files. In case of masks, which include a large number of objects, this could be time-consuming operation. The AS32 program permits buffering the masks in the operation memory of a computer. It means that buffered masks are read once (at the start of the system) and opening of these masks is performed without any disk operation.

10.2.3. Operator's Actions

PANEL
EXIT

See: 16.2. Description of Actions.

10.3. Using the Printer

AS32 program has built-in module of printer management. It permits execution of printouts in parallel to usual system operation. Printing does not influence on refreshing of visual masks and execution of other operations connected with operation of application.

Printouts may be performed on any printer installed in the system. No specific type of a printer is required. Text printouts (paper copy of reports, event log paper copy) require printers with font built-in in the hardware, which permit to print national characters and characters of frames used in reports. For the polish version it is LATIN II.

In **asix** system various information may be printed. They may be divided generally into text and graphic printouts.

The basic feature of printer module is the possibility to make queues of printouts. This permits the operator to queue a lot of printouts without waiting for real action. Handling module ensures that all printouts will be performed. AS32 program permits also immediate printout of events connected with process alarms (current printout of alarms is declared in ***Alarms printing*** parameter in application configuration file with use of Architect program: Architect > *Fields and Computers* > *Printout* module > *Alarm printing/Options* parameter). If printer is not busy - the handling module prints the alarms. If, however, other order appears - printout of alarm is interrupted. Consecutive coming alarms are stored in a buffer (cache), and after printer is released they will be printed immediately. After completion of print task the new page (Form Feed) sign is send to the printer to start new task on the new page.

The system operator has possibility to influence the printout system by means of special window opened with the command TOOLS.PRINTER_MANAGER from menu of the Designer Window or the Control Panel.

Printer operation window has the form as follows:

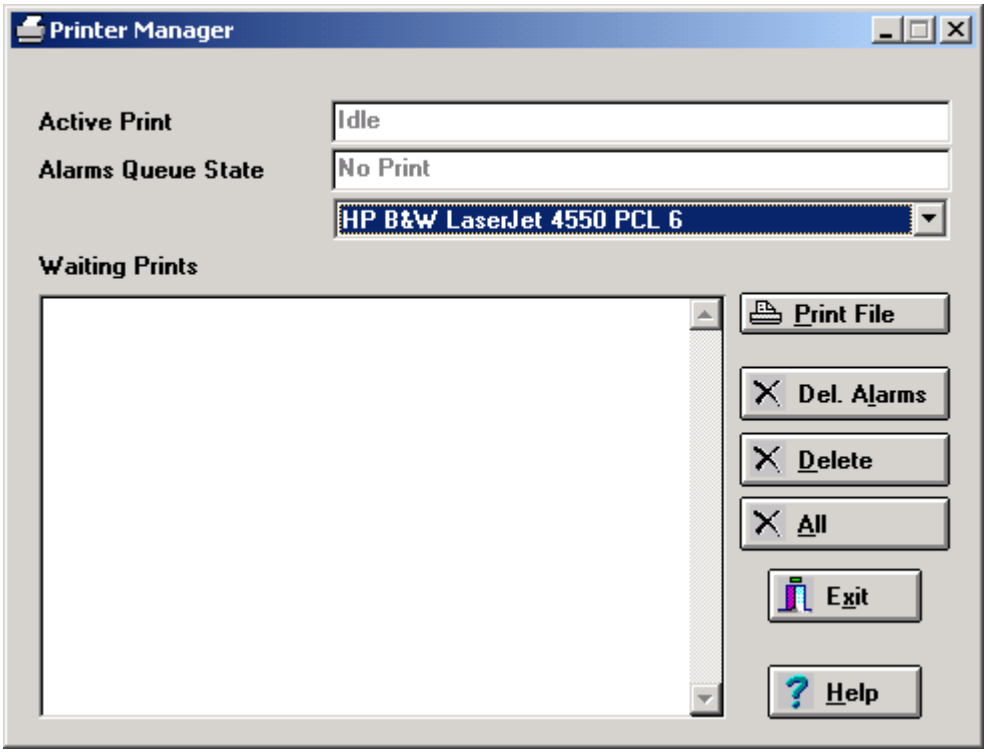


Figure 10-2. 'Printer Manager' Window.

The separate control fields have the following information:

Active Print - description of printout executed at present is specified in the field

Alarms Queue State - a field used in case, when in application system printing of alarms is executed currently. This field specifies the state of the buffer, which stores descriptions of alarm events. The following values are possible:

- *empty* - no alarm waits for printing.
- *used* - the buffer is partially filled. Normal situation if printer was busy with other printouts or temporary inflow of large number of alarms occurred. The system will print the alarms at the earliest opportunity
- *full* - the buffer is entirely filled. Coming alarm events cause loss of information of a printout. Such situation is typical when printer was switched off for long period of time. The second reason could be execution of other long printouts. If this situation repeats, it could be advisable to increase the capacity of the buffer in item ALARM_LISTING of initialization file.

Waiting Prints - descriptions of all ordered print jobs are shown on this list, in such order how they will be printed.

Below, types of printouts executed in **asix** system together with their descriptions appearing on waiting printout list are given:

current alarms - printing of alarms in immediate mode,
historical alarms printout - printing of alarms, which have been ordered by system operator on historical alarm mask,
report printout - printout of created report,
printout of graphic window - graphic printout of synoptic mask ordered by means of a command TOOLS. GRAPHICS_PRINT from menu Designer Window.
graphic screen dump - graphic printout (dump) of present content of the screen caused by pressing key combination *Alt-F5*
file_name - printout of any file, which had been initiated from printer window.

Despite of control functions, printer window has also possibilities to control printout execution. You may use buttons located to the right of the window. Their meaning is as follows:

Delete alarms - clears alarm buffer used while current alarm, printout is made.
Delete - permits selective deleting of one of waiting print tasks. Before pressing the key, you should select a print job from the list of waiting tasks.
All - allows deleting all waiting print jobs as well as interruption of current task.
Print file - pressing this button will cause opening of that window in which operator may select any file stored on disk. The file shall be printed by means of a printer selected in the installed printers list.
Exit - closing the handling window.

REMARK *List of waiting print jobs relates to all printout being executed by AS32 program regardless of a printer for which they are provided. The items are deleted of the list while they are transferred for executing by Windows system (they may be additionally queued before they are really printed).*

10.3.1. Dump of Screen Contents

Operator initializes the printouts by pressing *ALT-F5 key.* The following dialogue window appears:

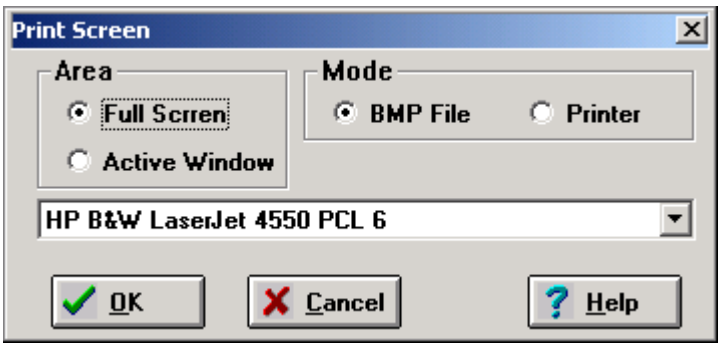


Figure 10-3. 'Print Screen' Window.

In this window the operator should set parameters, in accordance to which the printout is to be made:
an area, to which the operation relates. Printout of the full screen or the active mask is possible.
printout (dump) format. The following formats are possible:

Bitmap - graphic format of BMP type file. Printout in this format may only be made to the disk file.

Printer - printout selected for printing

printer port; printout on any, installed in Windows 2000/NT 4.0 systems printer is possible; at the first usage the default system printer is suggested.

In configuration file, in ***Dump parameters*** parameter you may define default parameters of the dump:

Architect program > *Fields and Computers* > *Printout* module > *Screen printout* tab

The above described window will be displayed with pre-determined default parameters. Before printout is made, an operator may still change the parameters. In ***Automatic dump*** parameter you may demand that the dump be made immediately after pressing *ALT-F5* key. Printout shall be performed in accordance with default settings and, in place of parameter window, the message window informing only on the fact that the dump is being made, shall be displayed. This message shall be automatically removed after 1 second. Below, an example of message window is shown. Dump of the screen without any signalling is also possible.

Figure 10-4. The Message on Screen Dump.

Beginning of dump is performed while parameters are determined. For period of this action, refreshing of synoptic mask is stopped. Completion of dump is indicated with sound signal, after which the system returns to normal state.

Dump files are located in working directory of the program declared in ***Work directory*** parameter in application configuration file with use of Architect:

Architect program > *Fields and Computers* > *Miscellaneous* module > *Application* tab

Filenames are given in accordance with the pattern: *screen???.bmp* for dumps of the full screen, *window???.bmp* for dumps of active window. In place of *???*, in the name of file the consecutive numbers of dumps are inserted.

10.4. Time Setting

System time updating in **asix** system is performed by means of special window. This window may be opened with OPTIONS.TIME_CHANGE command in the Designer Window, in the Control Panel or with of NEW_TIME operator's action.

Time updating window is presented below:

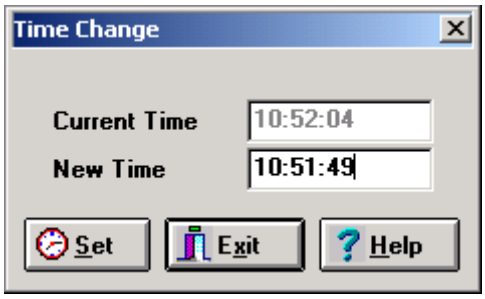


Figure 10-5. 'Time Change' Window.

In the box *Current time*, permanently updated system time is shown. To change time, operator has to enter new time in the box *New time* and click on *Set* button.

AS32 program permits limitation of period of time, by which you may execute a single modification. This period is defined in configuration file in ***Time change limit*** parameter with use of Architect program: Architect > *Fields and Computers* > *Miscellaneous* module > *Misc* tab). Limit of changing is provided to protect against accidental large changes of time, which could make archiving of data and alarms with improper time. Typical limitation is one-hour period, which allows for current corrections of time as well as transition between winter and summer time. Entire lock of time change is also possible.

10.4.1. Operator's Actions

NEW_TIME

See: 16.2. Description of Actions.

10.5. Demonstration Sequences

AS32 program provides one more mechanism used for automatic restoring of previously saved operations. They are so called demo sequences, which are to be used in demo versions of applications. They permits running the application, the operation of which is controlled by means of earlier recorded (by system designer) sequences of mouse movements as well as strikes of keyboard keys.

The general principle of demo sequences application are similar to rules of macro-operations.

Below, the specific features of demo sequences are described:

- there is possibility to define only one sequence. It is saved in file *replayw.dem*.
- demo sequences may be carried-out either in designer or application mode.
- in process recording and playing, time depending relationships are saved.
- recording begins with a OPTIONS.DEMO.RECORD command of the Designer Window or the Control Panel
- recording is completed by means of *Alt-F12* key.
- playing process is commenced with the command OPTIONS.DEMO.REPLAY from the Designer Window or the Control Panel. There is possibility of automatic initiation of playing at the moment when application is started, by means of appropriate definition of item SEQUENCE_DEMO in *asix.ini* file.
- in common mode of operation, using demo sequence is locked. Mechanism must be specially unlocked in item SEQUENCE_DEMO in *asix.ini* file.

Depressing *Ctrl-Esc* keys breaks execution of demo sequence. If given sequence is not broken after their completion, automatic playing shall be restarted from the beginning. Due to these reasons designer has to complete recording of the sequence using such a method to make possible their correct restart (the condition of program at the moment of beginning and the end of recording the macro must be sufficiently compatible).

10.6. Screen Keyboard Mechanism

Screen keyboard mechanism (active in the application runtime mode) provides a possibility to enter texts to the editing fields of the dialogue boxes and the editing fields of NUMBER and STRING objects with use of screen keyboard. This mechanism is designed to support application operating on touchscreen panels.

The keyboard is activated after clicking in the field or object area.

Screen keyboard mechanism is activated with use of ***Screen keyboard*** parameter of application configuration file, declared in:

Architect program > *Fields and Computers* > *Masks* module > *Screen keyboard* tab

There are two types of keyboards: for text edition and numerical one.

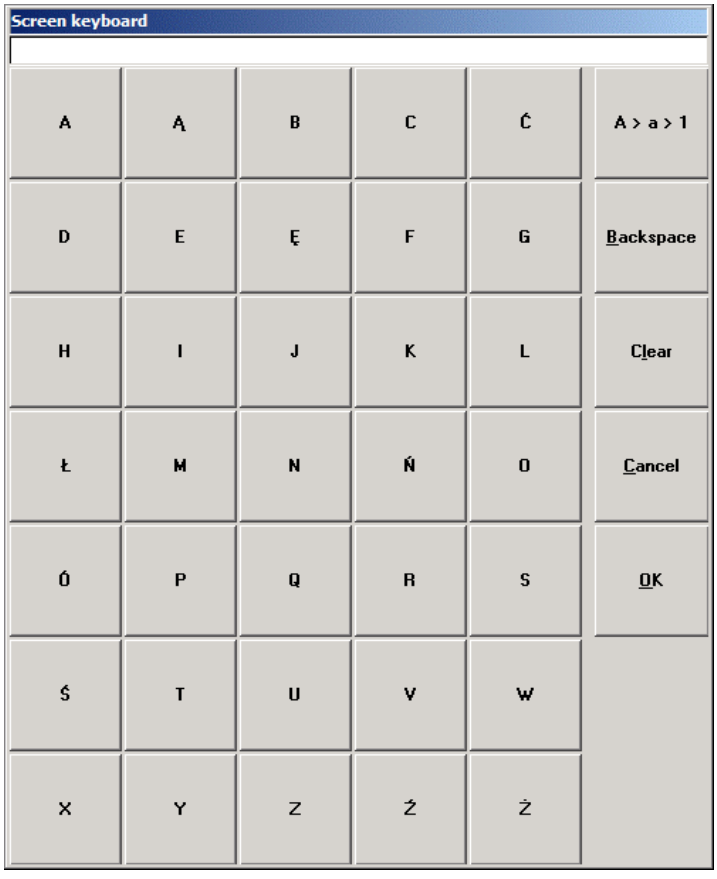


Figure 10-1 .Screen keyboard.

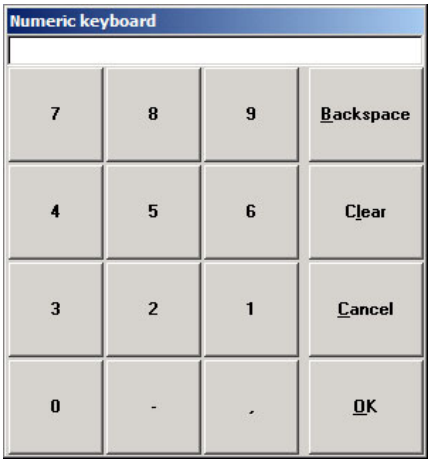


Figure 10-2. Numeric keyboard.

11. Variable Tables

Variable Tables enable you to create windows designed to inspect the current values of ASMEN variables. You can access the Variable Tables from menu of designer window or in run-time mode by using the *TABLE* operator action. You define the table interactively by choosing the options and parameters.

Variable Tables may be used only if you define the Variable Database before.

Variable Tables enable you to display as well values of analog variables as strings assigned to binary signals.

11.1. Opening the Table

In run-time mode the Variable Tables created earlier may be opened with use of TABLE operator action.

In menu of designer window you can find TABLE command. It is designed to open empty variable table.

11.2. Structure of Table Definition File

Data defining the table are stored in text files with ***.TBL** extension, having the structure of Windows .INI files.

The **[TABLE]** section in *.tbl file defines global parameters for windows of the table.

☑ ***TITLE= <string>***

Meaning - title of the table to be displayed.
Definition - manual.

☑ ***POSITION= x, y***

Meaning - position of table window on the screen defined as coordinates of its left top corner.
Definition - automatic.

☑ ***WIDTH = -1|<number>***

Meaning - width in pixels of table window. When value –1 is declared, width will be defined automatically basing on font type, column width and windows parameters.
Definition - automatic.

HEIGHT = -1|<number>

Meaning - hight in pixels of table window. When value –1 is declared, hight will be defined automatically basing on font type, number of rows and windows parameters.
Definition - automatic.

NUMBEROFCOLUMNS = <number>

Meaning - declaration of number of columns to be displayed. In current version, the number of column is constant and equal 4.
Definition - automatic

VALUECOLUMNNO = <number>

Meaning - defines a column number where the value of variable is to be displayed. In current version, the value 3 is accepted only.
Definition - automatic.

NUMBEROFROWS = <number>

Meaning - number of rows of the table to be displayed.
Definition - automatic.

FONT = <name>, <dimension>, [<font attributes>], <coding page>

Meaning - definition of name and parameters of font used to fill the table.
Definition - automatic.

DISPLACEMENT = <number>

Meaning - definition of displacement in percent of text according to overall position of. the table.
Default value - 30 %.
Definition - automatic.

FORMAT = <format>

Meaning - definition of displaying format of the variable compatible to that of numeric variables in **asix** system. It is applied when the format of presentation of value of measurement was not defined in Variable Database.
Definition - automatic.

BAR = 0|1

Meaning - item defining whether description bar of table window should be visible.

Definition - automatic.

GRID = 0|1

Meaning - item defining whether grid in internal part of the table, separating individual items is to be displayed.
Definition - automatic.

STRIPES = 0|1

Meaning - item defining whether the subsequent items of the table are to be displayed alternatively in colors of background and bar. Using this option improves the readability of table contents.
Definition - automatic.

ERRORFORMAT = <format>

Meaning - format of presentation of variable value; used when the value is provided with error status.
Definition - automatic.

The **[COLOURS]** section in *.tbl file defines colors of presentation for all items of the table. Each color is defined with 3 RGB components. Convention used for color definition is compatible with that used in initiating files of the AS program.

BACKGROUND= R, G, B

Meaning - definition of background color of the table.
Definition - automatic.

FONT = R, G, B

Meaning - definition of font color of the table.
Definition - automatic.

BLINKING = 0|1

Meaning - definition of blinking attribute of font of the table.
Definition - automatic.

BARBACKGROUND = R, G, B

Meaning - definition of alternative color of background of the table
Definition - automatic.

BARFONT = R, G, B

Meaning - definition of font color in bar of alternative background color of the table
Definition - automatic.

BARBLINKING = 0|1

Meaning - definition of blinking attribute of alternative color of font of the table.
Definition - automatic.

LIMITHHBACKGROUND = R, G, B

LIMITHBACKGROUND = R, G, B
LIMITLTBACKGROUND = R, G, B
LIMITLLTBACKGROUND = R, G, B

Meaning - definition of background color of the table in case where the limits are violated.
Definition - automatic.

LIMITHHFONT = R, G, B

LIMITHFONT = R, G, B
LIMITLLFONT = R, G, B
LIMITLLFONT = R, G, B

Meaning - definition of font color of the table in case where the limits are violated.
Definition - automatic.

LIMITHHBLINKING = 0|1

LIMITHBLINKING = 0|1
LIMITLBLINKING = 0|1
LIMITLLBLINKING = 0|1

Meaning - definition of blinking attribute of font color of the table in case where limits are violated.
Definition - automatic.

ERRORBACKGROUND = R, G, B

Meaning - definition of background color of the table for measurements provided with error status.
Definition - automatic.

ERRORFONT = R, G, B

Meaning - definition of font color of the table for measurements provided with error status.
Definition - automatic.

☑ ***ERRORBLINKING = 0|1***

Meaning - definition of blinking attribute of background color of the table for measurements provided with error status.
Definition - automatic.

The **[COLUMNS]** section defines the name and width (in characters) of columns of the table.

☑ ***<column_no> = <name>, <width>***

Meaning - definition of column header and its width.
Definition - manual.

The **[ROWS]** section in *.tbl file defines contents of rows of the table. Defining the name of variable enables retrieving from Variable Database such information as description, unit, presentation format, limits. Using the name and ASMEN program you can retrieve the current value of measuring variable. When the name is not defined, the row of table will be left empty.

☑ ***<row_no> = [<name of variable>]***

Meaning - indication of variable in Variable Database. Information on the indicated variable will be displayed in row with specified number.
Definition - manual.

EXAMPLE

Contents of example *.TBL table definition file.

```
[Table]
Title=Steam and Gases Flow - Boiler no 15
Position=60,25
NumberofColumns=4
NumberofRows=31
ValueColumnNo=3
Font=MS Sans Serif,13,[],EastEurope
Format=%7.2f
Bar=1
Grid=0
Stripe=1
ErrorFormat=
Width=-1
Hight=-1
Displacement=30
```

```
[Colors]
Background=192,192,192
Font=0,0,0
Stripe=175,175,175
LimitHHBackground=255,0,0
LimitHHFont=255,255,255
LimitHHBlinking=1
LimitHBackground=255,0,0
LimitHFont=255,255,0
LimitLBackground=0,0,160
LimitLFont=255,255,0
LimitLLBackground=0,0,160
LimitLLFont=255,255,255
LimitLLBlinking=1
ErrorBackground=255,255,191
ErrorFont=255,0,0
Blinking=0
StripeFont=0,0,0
StripeBlinking=0
LimitHHDefautltBackground=0
LimitHDefautltBackground=0
LimitH=0
LimitLDefautltBackground=0
LimitLBlinking=0
LimitLLDefautltBackground=0
ErrorDefautltBackground=0
ErrorBlinking=0

[Columns],15
2=Description,30
3=Value,7
4=Unit,7

[Rows]
1=K15_SPV_Para
2=K15_CRR_Para
3=K15_PVL_Para
4=K15_STA_Para
5=K15_Para
6=K15_LV_Para
7=K15_TRN_Para
8=
9=K15_CLSS_Para
10=K15_IPVL_Para
11=K15_INTG_Para
12=K15_ILV_Para
13=K15_DAVG_Para
14=K15_DTRN_Para
15=K15_MAVG_Para
16=
17=K15_SPV_Vss
18=K15_CRR_Vss
19=K15_PVL_Vss
20=K15_STA_Vss
21=K15_Vss
22=K15_LV_Vss
```

```
23=K15_TRN_Vss
24=
25=K15_CLSS_Vss
26=K15_IPVL_Vss
27=K15_INTG_Vss
28=K15_ILV_Vss
29=K15_DAVG_Vss
30=K15_DTRN_Vss
31=K15_MAVG_Vss
```

11.3. User Interface

Variable Tables take form of the window presented below:

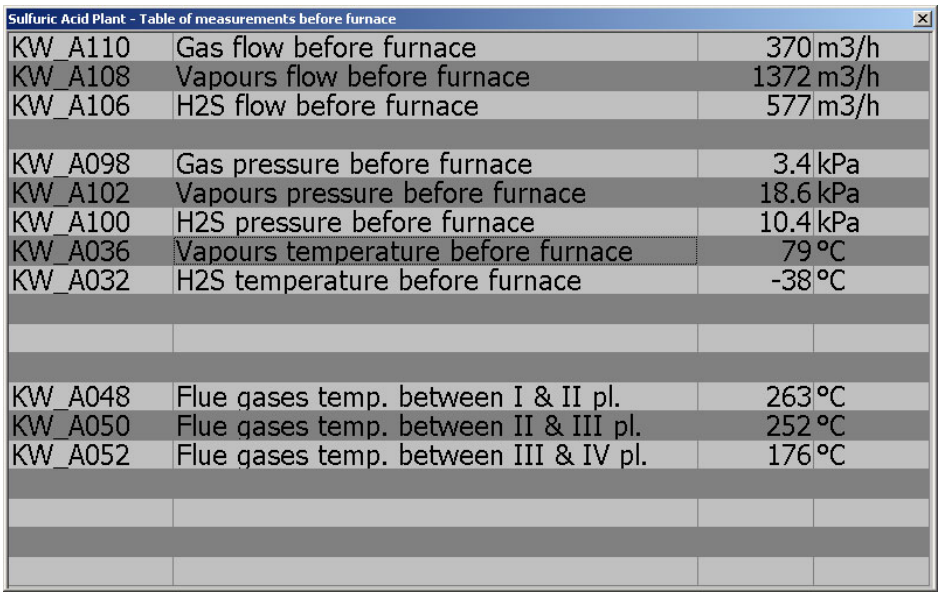


Figure 11-1. Table of Measurements.

It includes four columns: variable name, variable description, variable value, and unit respectively. Variable name is specified by designer. Description and unit are retrieved from Variable Database and the values are retrieved on-line from ASMEN module. Header of the table window contains the name of table.

In order to select a variable for presentation double-click with mouse in the row of table. This causes opening of window including the list of variables. The window with list of variables may be also opened with use of context menu of the table.

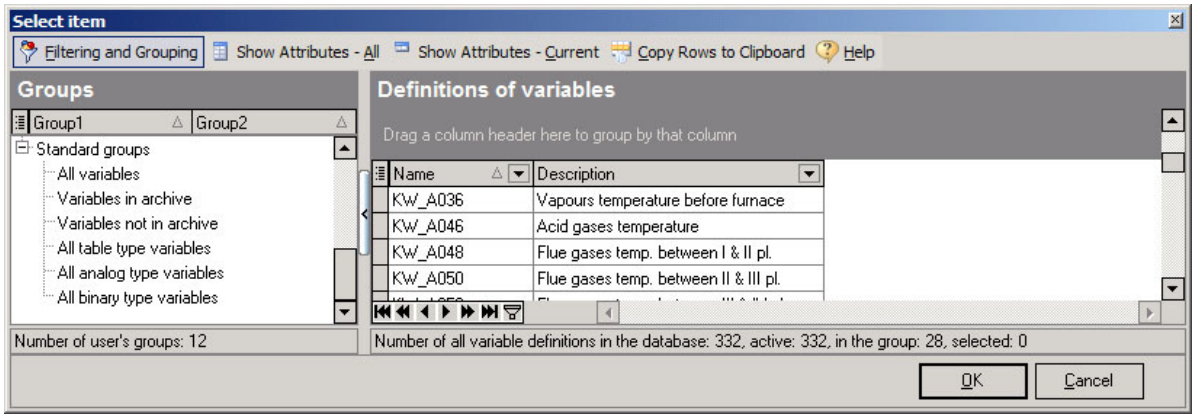


Figure 11-2. 'Select item' Window.

Context menu of table, activated with right button of the mouse may take two forms depending on mode of operation of AS program, a namely: run-time mode and designer mode.

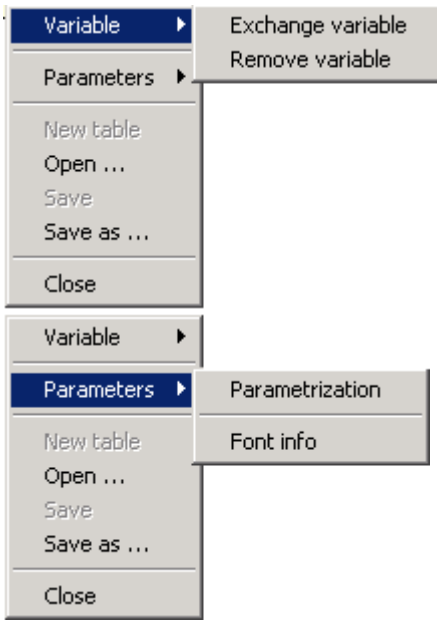


Figure 11-3. Context-Sensitive Menu of the Measurement Table.

In run-time mode, the context menu enables you to replace the variable in row of the table (with use of variable list window), to delete the variable from the row of table, to save the modifications and to close the table window.

In designer mode, the context menu enables you to define you the table window, to display the detailed information on applied font, to open another table window and to save the current window under new name.

Defining the table window is effectuated with the use of the following window:

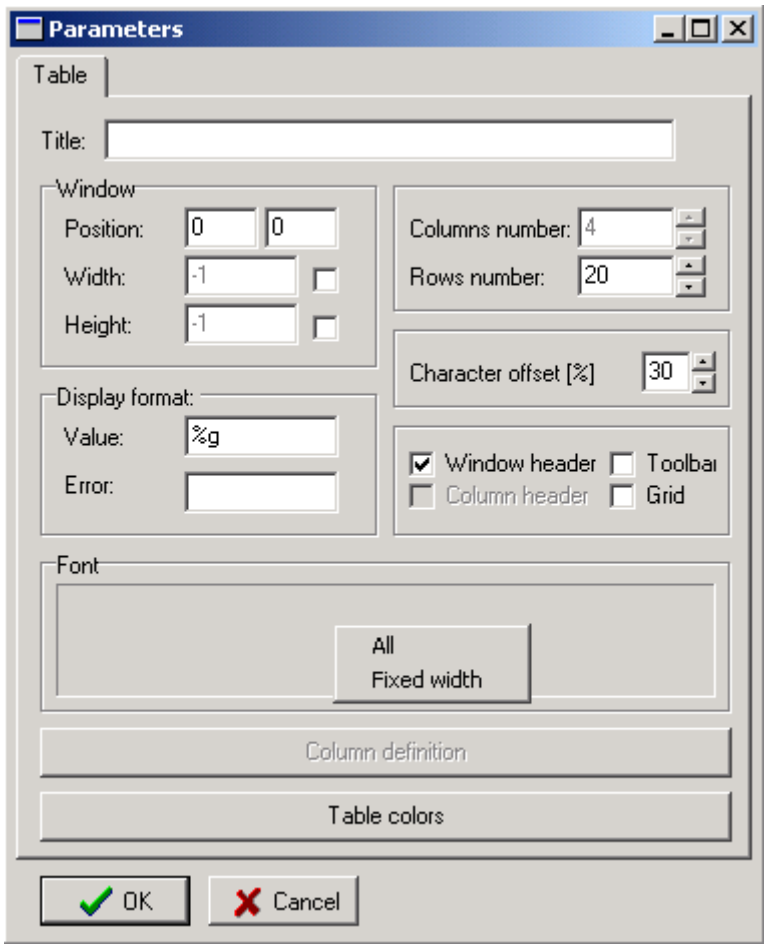


Figure 11-4. The Window for the Table Parameterization.

It enables you to modify the following parameters:

- Title of the table;
- Position of the screen;
- Width of window; Value -1 denotes that width of windows will be defined in automatic way;
- Hight of window; Value -1 denotes that height of windows will be defined in automatic way;
- Number of columns of the table; In current version this parameter takes constant value 4;
- Number of rows of the table;
- Displacement of character;
- Default presentation format of the variable;
- Presentation format of error values;
- Options of presentation of window header, bars and grid;
- Font;
- Color of presentation.

In frame *Font* an example text with current font is displayed. With use of context menu you may change font or its attributes. You may display the font selection window for all fonts registered in system or for fonts of constant width only.

When you click on *Table Colors* button, the window appears that enables you to define colors of all items of the table window. Further frames include test text displayed according your settings for:

- Critical upper limits (HH);
- Upper limits (H);
- Main text;
- Table bar;
- Lower limits (L);
- Critical lower limits (LL);
- Erroneous measuring values.

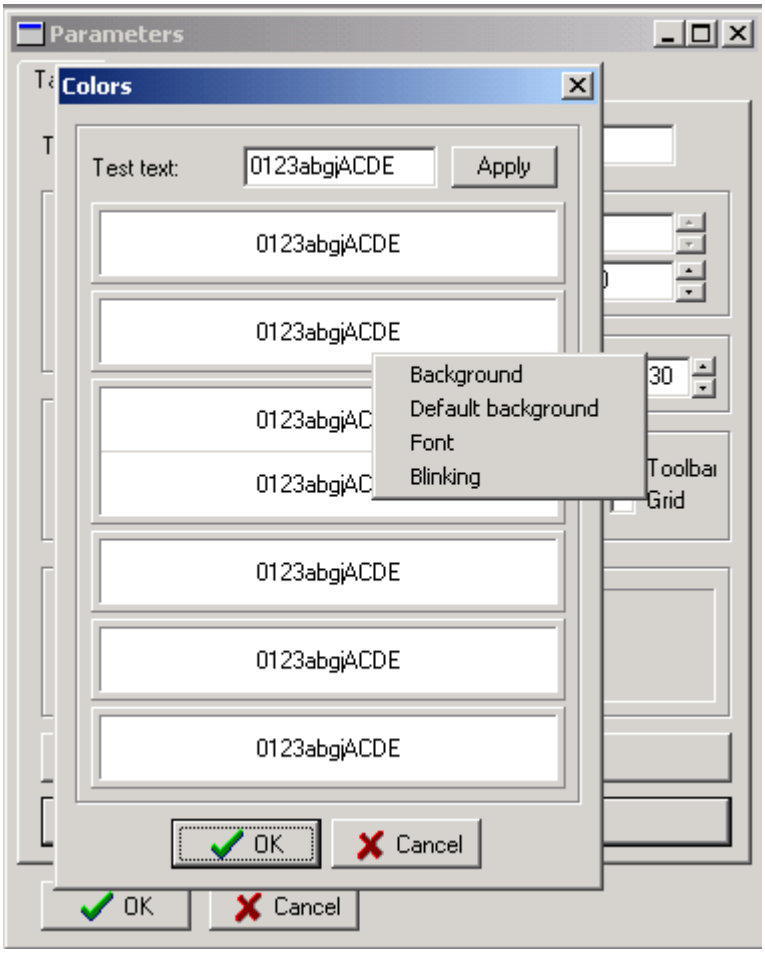


Figure 11-5. The Window of the Table Colors.

With use of the context menu of the individual frames you may modify the following attributes:

- Background color; for limits and errors you may use default background i.e. main text color or bar color.
- Font color.
- Blinking of value

In the context menu (designer mode) you can display window including detailed information on selected font. The window is active until closed by operator.

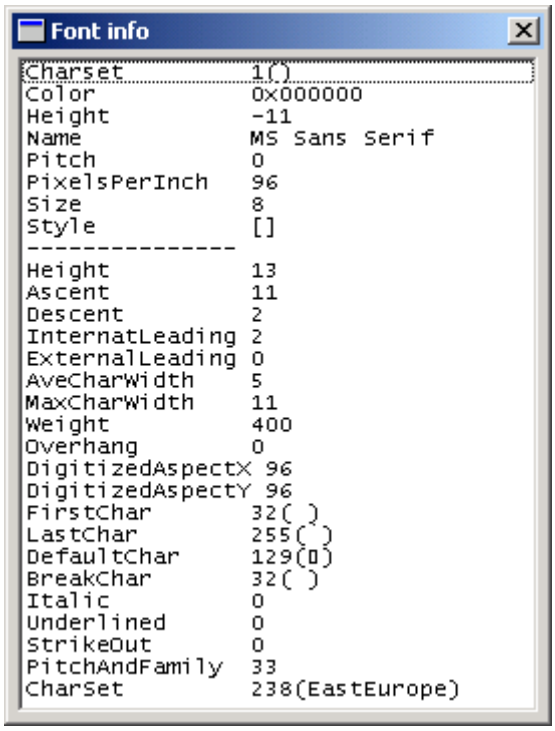


Figure 11-6. The Window of Font Information.

11.4. Binary Signals

In order to display in Variable Tables the text corresponding to the value of variable, you should place the corresponding format string that begins with %@ characters to the format field.

Example format string:

%@ 0="OFF" ; 1="ON"

- for value of variable = 0, text OFF appears
- for value of variable = 1, text ON appears

Conversion functions for these variables should return as a result an integer number, e.g. NOTHING, NOTHING_BYTE, NOTHING_INT

In file schemat.txt defining the Variable Database you should change dimension of the Format field in correspondence to your needs, taking into account dimension of the longest format string.

12. Alarm System

asix has built-in flexible, configurable System of alarm management, which allows to fulfill entirely the needs connected with complex handling of failure situations and events in engineering process. The basic features of this system are as follows:

- long-term history of alarm log limited only with disk capacity,
- separate log of current alarms,
- automatic retrieval of alarm conditions after system restart,
- possibility to define 65535 various alarms,
- partition of alarms into 5 types:
  - system alarms,
  - messages,
  - warnings,
  - alarms,
  - critical alarms;
- partition of defined alarms into unlimited number of groups,
- sound signalling of alarm occurrence,
- millisecond alarm resolution,
- possibility of on-line alarm printing from specified period of time,
- two type of masks provided for displaying alarm conditions:
  - active alarm mask,
  - historical alarm mask;
- possibility of selection of presented alarms according to 5 criteria:
  - time,
  - text,
  - type,
  - status,
  - group,
  - numbers;
- saving the alarm selection status to the file; this status will be set when you read later the contents of this file for presentation;
- possibility to exclude selected alarms from service;
- saving the disabling status to the file; this status will be set when you read later the contents o this file for presentation;
- possibility of filtering of selected alarms;
- mechanism of alarm acknowledge;
- possibility to include values of process variables into texts alarm message;
- possibility to connect an alarm with synoptic mask;
- strategy of alarm recognition:
  - map of alarm bits,
  - test of exceeded limit values;
- synchronous operation of several operators:
  - handshaking of alarm system state during restart of the operator station,
  - acknowledgement of alarm is sent to all stations,
  - synchronization of disabling and filters;
- browsing of historical alarms occurring in other, network-connected stations;
- possibility to create a backup copy of the historical log files;

- cooperation with PLC controllers in order to switch off alarms and to control the sound signalling.

In general, the alarm system may be divided into two parts: alarm manager, which is responsible for detecting, saving and defining the alarms, as well as the visualization part, composed of two types of alarm masks (for current and historical alarms).

Parameters of alarm system operation are set up with use of Architect program.

See more detaled information in:
Architect user's manual, chapter *3.9. Configuration of alarm system*.

12.1. Configuration of Alarm Manager

Configuration of alarm system manager is based on appropriate definition of items in initialization file as well as on development of text file including additional information. The most of items related to alarm are located in section ALARMS_SYSTEM. Additional text files are as follows: files of alarm definitions, files of group definitions, files of limit definitions.

The current alarms as well as historical ones are stored in disk files. Unless otherwise defined, alarm files are stored in subdirectory *alarms* of starting directory. All active alarms are stored in the file named *alarms.act.* Alarm history is stored in set of files, which names are derived from standard *al??????.log* type. In such type of file the alarms belonging to one day are stored. The date of this day replace *??????* characters, specifying day, month and year, respectively. The number of stored day history files may be limited by means of proper adjustment of ***Limitation of alarm files storage*** parameter of application configuration file, declared with use of Architect:

Architect program > *Fields and Computers* > *Alarms System* module > *Archive* tab

12.1.1. Files of Alarm Definitions

Any alarm, which is to be used in **asix** system, must be previously defined. Definitions of alarms are located by default in *alarm.def* file. It is possible to use another file specifying it with use of ***Names of files where alarms are defined*** parameter (Architect > *Fields and Computers* > *Alarms System* module > *Files* tab). Description of alarms may also be located in several definition files. Internally, alarms are identified by their numbers. You may define 65535 various alarms. There is no necessity for sequence numbering of alarms. For instance, numbering of one group of alarm begins from 1000, and the second one (of other origin) from 30000. You have to remember that changing the alarm numbering in working system causes the necessity to erase the alarm archive files.

One of five available types (basically four, because the system alarm type relates to internal software state of **asix** system and cannot be applied to alarms connected to the state of a process under control) is assigned to each alarm. Type of alarm cause that alarms are displayed on alarm masks in various colors. In the moment of alarm occurrence different sound signals are also generated. Types of alarms are used as one of selection criteria too.

Each alarm has assigned the text corresponding to it, which is displayed on alarm masks as well as printed on a printer. It is possible to connect two various texts with an alarm: one is used in the moment of alarm occurrence, whereas the second one is used when the reason of alarm was withdrawn. In text of alarm you may include formatting characters in function of *sprintf* of C language. It permits to insert to the alarm description the parameter values related to detected alarm. However, it is required that handler, which recognizes the reason of alarm, could transfer these additional information together with number of alarm. At present, only limit alarm handler can transfer the value of variable and limits in the moment of occurrence and withdrawal of alarm.

The alarm system allows to split all defined alarm into unlimited number of groups. Splitting into groups is used as one of criterion of alarm selection. To define an alarm you have to use 4-character group identifier. Additionally, in group definition file, you have to define the full names of groups, which shall be displayed in window of alarm selection. System operator is not obliged to know artificial group identifiers; he selects from the list of texts describing defined groups.

An exception is created automatically group of alarms number 0, to which belong all internal events of AS program. It allows selecting those alarms with group criteria for presentation. In an alarms selection criteria window, in the groups section a group called **System State Events** should be selected.

There is also possibility to connect alarms with process masks. Name of associated mask is entered in alarm definition. For so defined alarm, the system operator may, by pointing the alarm on alarm mask, and clicking on the *Mask* button cause immediate passing to associated synoptic mask, showing a part of process installation, related to the alarm.

Alarm definition file is a text file. Content of this is composed of lines, each one describes one alarm. Blank lines are ignored. A single description is composed of fields separated with commas. Spaces at the beginning and at the end are ignored. All fields have assigned the default values. If a field is not set - insert comma characters only. If all fields are not set until the end of line, commas may be neglected. The maximum length of a line is 200 characters.

The structure of the line describing the alarm is as follows:

[alarm_number],[alarm_type],[alarm_beginning_text],[alarm_end_text],[group_name],[mask_name],[PRINT|NO_PRINT]

where:

alarm number	- omitting denotes an alarm definition of the next number in relation to the previous alarm (the first alarm in the file has number 1);
alarm type	- declaration of alarm type; texts used to determine the types are as follows: SYSTEM, MESSAGE, WARNING, ALARM, IMPORTANT, IGNORED; it is

	sufficient to specify two first letters; type ALARM is accepted by default; type IGNORED enables to exclude some alarms of servicing, at the stage of designing; (for instance, ignoring of some positions in alarm map of bit strategy);
alarm beginning text	- a text displayed in the moment of alarm detection;
alarm end text	- a text displayed for alarms, which have been completed; the lack of specification for this text denotes that the same text as for beginning of alarm is to be used;
group name	- 4-character identifier of a group, for which the alarm belongs;
mask name	- name of suggested mask;
[PRINT|NO_PRINT]	- enabling|disabling on-line alarms printing.

12.1.2. Files of Group Definitions

The definition file of alarm groups is a text file. The same principles are used as for alarm definition file. Definitions of alarm groups are located by default in file *group.def*. It is possible to use a file (several files) of other name defining the ***Names of files where alarm groups are defined*** parameter (Architect > *Fields and Computers* > *Alarms System* module > *Files* tab).

The structure of the line describing the group of alarms is as follows:

group_identifier,[group_description],[PRINT|NO_PRINT]

where:

group identifier	- maximum 4-character identifier of a group, used for identification of groups in the alarms definitions file;
group description	- any text describing the group; this text shall be displayed in the list of groups of alarm selection window; lack of text will cause usage of group identifier in selection list;
[PRINT|NO_PRINT]	- enabling|disabling on-line alarms printing(instantly) for a specific group.

12.1.3. Control of Cumulative Alarm Group State

The mechanism of control of cumulative alarm group state allows displaying state of activity of alarms belonging to a common group. This mechanism activation for selected group consists in defining the variable, named in accordance with the below rule, in the NONE channel:

__AGR_groupidentifier__

The group identifier should be compatible with one declared in the file of group definition.

If the group control variable is created, the alarm system will automatically refresh this variable value in accordance with the below rules:
- value 0 means lack of active alarm in the group,
- value 1 means that there are active alarms in the group, but they are all confirmed,
- value 2 means that there is at least one un-confirmed alarm in the group.

The group control variable may be used in other objects to determine displaying attributes.

12.1.4. Selection of Alarms Printed in the On-line Mode

Declaration of the alarm printing, can be made in two ways:
- In the alarm group definition a PRINT parameter can be added, defining that all alarms from this group should be printed. Missing the parameter (or NO_PRINT) means, that the group will not be printed.
- At the end of individual alarm definition a PRINT or NO_PRINT parameter can be added. Individual specification takes precedence over group specification. Missing parameter means accepting the group setting or not printing for the off-group alarms.

If above print specifications aren't used, it is assumed that all alarms have the printing permission.

The ***Print system events*** parameter (declared for application configuration file with use of Architect > *Fields and Computers* > *Printout* module > *Alarm printing/Options* tab) specifies if the system events should be printed.

12.1.5. Strategies of Alarm Detections

Bit Strategy of Alarm Detection
Limit Strategy of Alarm Detection
    Examples
Active Strategy of Alarm Detection
Buffered Strategy of Alarm Detection
    Variable Definition in asix Database
OPC Alarm Strategy

Besides defining the alarms, it is still necessary to determine the method of alarm detection. There are five strategies of alarm detection in **asix** system.

Bit strategy consists in checking the bit state in bit map of alarms, created by program of a PLC.

Strategy of limit control consists in periodical comparison the value of selected process variable with specified limits.

Active strategy consists in exchanging the list of alarms with the controller. List of alarms is sorted chronologically with time resolution higher than 1 sec.

Buffered alarm strategy allows detecting alarms with millisecond resolution and to assign them a timestamp by the controller exactly in the moment of event occurrence.

OPC alarm strategy allows the alarm information to be sent from OPC Alarm and Event Server to **asix** system with OPC alarm driver (that is used as a client of any OPC Alarm and Event Server).

See detailed information on configuration of alarm detection strategy: Architect user's manual, chapter *3.9.5. Definition of alarms strategy*.

Bitmap Strategy of Alarm Detection

Parameters of bit strategy are set by suitably defined BITMAP_STRATEGY items. It is possible to handle several bit maps of alarms at the same time, each being defined in a separate item. Two basic parameters of the bit map are the name of ASMEN's variable used to read the map from a controller, and the number of the first alarm in the map. You have to ensure the compatibility of this number with arrangement of alarms in the definition file. Every successive bit in the bit map corresponds with alarm numbers increased by one. If bit in alarm map is set to 1 , the corresponding alarm is active. If is 0, it is considered the alarm is inactive. Amount of alarms located in a single map results from the size of ASMEN's process variable provided for reading the map.

Limit Strategy of Alarm Detection

Limit strategy parameters are defined with use of LIMIT_STRATEGY item and by creation of a text file (files) containing the descriptions of limits. Limit strategy is based on on-line monitoring values of process variables and their comparison with declared limits. No special action of controller software is required. You may define one of six comparisons. The value of process variable may be compared to literal value of limit fixed in advance or two different process variables. Basically, the limit strategy uses the analog variables. However, using "=" and "<>" (equal, different) conditions, you may verify value of binary variables.

Definition files of alarm limits are text files. The same principles are applied to them as to alarm definition files. The standard name is *limits.def*. The structure describing the alarm limit is as follows:

[alarm_number],variable_name,condition,limitation

where:

alarm number - omitting denotes that the next number is given in relation to the previous one (an alarm of the first limit in the file receives number 1);

variable name - name of ASMEN process variable, the value of which should be monitored;

condition - type of comparison operation. The following conditions are acceptable: <, <=, =, <>, >=, >;

limitation - a value or process variable name, to which monitored data should be compared; value may be of floating point or integer type; in case where variables are compared - you have to ensure the compatibility of their types.

Both the current value of monitored variable and limitation value is passed to alarm log by limit strategy handler in the moment of detection or completion of an alarm. They may be displayed in an alarm text on alarm masks by means of proper formatting characters in texts of the beginning and the end given specified in alarm definition.

EXAMPLES

Limit:	10,A1,>,A2
Assumption:	variable A1 and A2 are 16-bits numbers
Alarm text:	Temperature of %u housing is greater than critical maximum %u
Limit:	0,F1,>,3.4
Assumption:	Variable F1 is floating point character
Alarm text:	Temperature of housing %f is greater than critical maximum %f

The first formatting sequence in alarm text relates always to value of a variable, the second one - to limit value.

Active Strategy of Alarm Detection

The principle of handling consists in active transfer of information about alarm events from process controller to the computer. Using this strategy requires application of the transmission protocol, which allows to initiate the transfer by the controller, which should be provided with software module sending data in accordance with the protocol accepted by active strategy. It is defined by one or more items declared in *Alarms system* module of Architect program:

<buffer>, <synchro>, <alarm_number>,< id>

Parameters meaning:

<buffer> - name of the buffer used for data exchange;

<synchro> - name of the synchronizing variable;

<alarm_number> - number of the first alarm supported by this element of strategy;

<id> - identifier (network SINECL2, currently 1(SSNR =0) or 2 (SSNR =8)).

Buffered Strategy of Alarm Detection

This strategy enables high resolution of alarm detection in range of milliseconds, and assigning the time stamps for every alarm.

This strategy is equivalent to the active strategy. However, it doesn't require active data transfer from the controller. The controller fills its buffer with alarms. In the moment of appearance of any alarms in the buffer, the controllers sets a synchronizing variable. It's a signal for the **asix** station, that the data about new alarms are ready to be read. When the reading operation is over, the **asix** station resets the synchronizing variable. For the controller it means, that it can start filling the buffer with new alarms.

This strategy can be used with any communication protocol, capable of transferring the data in the form of 16-bit words arrays.

A cooperation with proper controller program is required for correct operation. Details, concerning this program, should be agreed with ASKOM company.

Configuring the Buffer Strategy is performed by entering the following items in *Alarms System* module of Architect program:

<buffer_variable>,<synchronizing_variable>,<alarm_variable>,<alarm_number>

Parameters meaning:

<*buffer_variable* > - name of ASMEN's buffer used for exchange alarm information; its size must be equal to the size of buffer in the controller;

<*synchronizing_variable* > - name of ASMEN's variable used for synchronizing the access to the buffer;

<*alarm_variable*> - name of ASMEN's variable used by the PC for making out the request of alarms map read – used during system initiation;

<*alarm_number*> - defines the number of alarm within the ranges of **asix** system, assigned to the first alarm detected by the declared strategy.

EXAMPLE

Declarations of buffer strategy declared in application configuration file with use of:

Architect program > *Fields and Computers* > *Alarms System* module > *Strategies/Buffer* tab

ALRMS1_BUFF, ALRMS1_SYNC, ALRMS1_MAP, 21

OPC Alarm Strategy

In OPC alarm strategy, the alarm information are sent from OPC Alarm and Event Server to **asix** system with OPC alarm driver (that is used as a client of any OPC Alarm and Event Server).

This strategy allows sending only active alarms. Data of alarm start and end time are sent with accuracy of one millisecond.

There is a notion of alarm in **asix** system – which signals some events (usually emergency) and has its start and end time.

OPC specification and OPC Alarm and Event Servers handle the notation of condition and sub-condition. The condition is a state of OPC server or one of its objects, e.g. "Alarm Level" condition may include sub-conditions like "Low Level", "High Level", "Very High Level", and so on. If a variable value increases above or below a defined level, then the variable changes over from a "normal" condition to one that has previously defined and is connected with the variable value.

Conditions and sub-conditions names are dependent on the server. If some object on the server (e.g. a variable) changes its condition or sub-condition, then the server generates an event which is received by the alarm client and translated into alarms in **asix** system.

For this purpose OPC Alarm and Event Server driver needs a file with translation of OPC conditions into alarms in **asix** system. The structure of this file is described in further part of the present chapter.

The objects (variables) on OPC server are divided into categories, the names of which are dependent on the server. When event connected with the variable of condition or sub-condition generating, the object that changes its state is named a source.

OPC alarm driver use

The line of the following syntax should be added in application configuration file with use of Architect program > *Fields and Computers* > *Alarms System* module > *OPC alarms* server:

ALOPC.dll, , <*server*>, < *al_opc_file*>

where:

<*server*>	- name of OPC Alarm and Event Server;
<*al_opc_file*>	- name of the file with translation of OPC conditions into alarms in the **asix** system.

EXAMPLE

ALOPC.dll, , TriangleMicroWorks.OPCAE, c:\asix\app\app.alarmyOPC

If alarms are to be displayed in **asix** system with accuracy of 10 milliseconds – the following parameter should be placed in application configuration file with use of Architect:

Architect program > *Fields and Computers* > *Alarms System* module > *Archive2* tab > ***Accuracy of time measure in alarm archives*** program

The format of the file with translation of OPC conditions into alarms in the asix system

The file structure:
- one row corresponds to one condition or sub-condition of OPC Alarm and Event Server;
- comment lines begin with semicolon (;);
- each line of the translation file has the form: <type>,<source>, <category>, <condition>, … other parameters depend on the type.

There are 6 types of alarms (ways of translating the OPC events into alarms in the **asix** system):

1. *alarm* - alarm in the **asix** system will be started after sub-condition (connected with it) activation or higher priority sub-condition activation (e.g.: HI_HI alarm activation will activate HI alarm automatically);
2. *alarm/switch* - alarm in the **asix** system will be started after activation of sub-condition connected with it;
3. *alarm/attribute/bool* - alarm in the **asix** system will be started after the sub-condition connected with it will be active or the event for an active sub-condition will occur, while the value transmitted by an attribute with a given ID will differ from 0.
4. *alarm/attribute/trip/start*;
5. *alarm/attribute/trip/impulse*;
6. *alarm/attribute/trip/durable* - alarm in the **asix** system will be started after the sub-condition connected with it will be active or the event for an active sub-condition will occur.

Types 4,5 and 6 differ from each other in the way of procedure when receiving a new event from OPC Server:

- for *start* type alarms – only the beginning of a new alarm (not the end of a current one) is generated;
- for *impulse* type alarms – the beginning and suddenly the end of a new alarm is generated;
- for *durable* type alarms – the end of a current alarm and then the beginning of a new one is generated.

In practice it means:

- for *start* type alarms:
  - in an active alarm dialog box the alarm value and date will change, but the alarm will stay still visible;
  - in a historical alarm dialog box only the alarm beginning will appear;

- for *impulse* type alarms:
  - in an active alarm dialog box nothing will appear;

- in a historical alarm dialog box the beginning and end of an alarm will appear;

- for *durable* type alarms:
  - in an active alarm dialog box the alarm value and date will change, but the alarm will stay still visible;
  - in a historical alarm dialog box the end of previous and the beginning of a new one will appear;

The format of the file with translation of OPC conditions into alarms in the **asix** system for *alarm* type conditions

The row of the translation file has the following form for *alarm* type conditions:

alarm, <*Source*>, <*Category*>, <*Condition*> [, <*Sub-condition*> <*asix alarm*>], * [, <*sub-condition*> <*asix alarm*>]

where:

<*source*>	- variable (object) name on the OPC server;
<*category*>	- name of condition category – a value specific for the given OPC server;
<*condition*>	- condition name – a value specific for the given OPC server;
<*sub-condition*>	- sub-condition name – a value specific for the given OPC server;
<*asix alarm*>	- alarm number in the **asix** system.

A couple of <*sub-condition*> and <*asix alarm*> parameters may repeat many times (usually two times).

Sub-condition names and alarm numbers are declared w.e.f. the alarm corresponding to the variable values from the highest to the lowest one.

The (*) sign is preceded by the names of H1 type sub-conditions (corresponding to the variable value higher than a normal one) and alarm numbers (in the **asix** system) corresponding to these sub-conditions. The names of sub-conditions of LO type (corresponding to the case when the variable value is lower than a normal one) together with alarm numbers (in the **asix** system) corresponding to these sub-conditions follow the (*) sign.

EXAMPLE

There is a variable x taking the values from 0 to 100, which has been configured on the OPC server in the following way:

Table 12-1. OPC Alarm Strategy - the Table for the Example.

Value of Variable x	**Condition***	**Sub-condition***
0 – 20	Level	Lo_Lo
20 – 40	Level	Lo
40- 60	-**	-**
60 – 80	Level	Hi
80 - 100	Level	Hi_Hi

* - condition and sub-condition names depend on the OPC server
** - normal state

We would like the alarms in the **asix** system of the following numbers to correspond to the declared conditions:

Lo_Lo – 10
Lo – 11
Hi – 20
Hi_Hi – 21

(None of alarms in the **asix** system corresponds to the normal state).

If the variable is named „Device1.Variable1", the category is named „Level Category" and the condition and sub-condition are named as in the above table, then the following line should be placed in the file with translation of OPC conditions into alarms in **asix** system.

Device1.Variable1, Level Category, Level, Hi_Hi 21, Hi 20, *, Lo 11, Lo_Lo 10

EXAMPLE

Numeric.AlSource, Limit, LIMIT_EXCEEDED, HI_HI 4, HI 3, *, LO 2, LO_LO 1

The format of the file with translation of OPC conditions into alarms in the **asix** system for *alarm/switch* conditions

The row of the translation file has the following form for single state conditions:

alarm/switch , <*Source*>, <*Category*>, <*Condition*> [, <*Sub-condition*> <*asix alarm*>]

It is also possible to declare a separate line for each condition:

alarm , <*Source*>, <*Category*>, <*Condition*>, <*Sub-condition 1*> <*asix alarm 1*>
alarm , <*Source*>, <*Category*>, <*Condition*>, <*Sub-condition 2*> <*asix alarm 2*>
alarm , <*Source*>, <*Category*>, <*Condition*>, <*Sub-condition 3*> <*asix alarm 3*>

where:

<*Source*>	- variable (object) name on the OPC server;
<*Category*>	- name of condition category – a value specific for the given OPC server;
<*condition*>	- condition name – a value specific for the given OPC server;
<*sub-condition*>	- sub-condition name – a value specific for the given OPC server;
<*asix alarm*>	- alarm number in the **asix** system;

EXAMPLE

alarm/attribute/trip/start , Numeric.AlSource, Limit, LIMIT_EXCEEDED, Default 255 401

where:
255 is an attribute ID the variable value is storaged in.

The format of the file with translation of OPC conditions into alarms in the **asix** system for *alarm/attribute/trip/start, alarm/attribute/trip/impulse, alarm/attribute/trip/durable* conditions

The row of the translation file has the following form for *alarm/attribute/trip/start, alarm/attribute/trip/impulse, alarm/attribute/trip/durable* conditions:

alarm/attribute/trip/start , *<source >, <category>, <condition> , <sub-condition > [<attribute ID>] <asix alarm>*
alarm/attribute/trip/impulse , *<source >, <category>, <condition> , <sub-condition > [<attribute ID>] <asix alarm>*
alarm/attribute/trip/durable , *<source >, <category>, <condition> , <sub-condition > [<attribute ID>] <asix alarm>*

where:

<*Source*>	- variable (object) name on the OPC server;
<*Category*>	- name of condition category – a value specific for the given OPC server;
<*condition*>	- condition name – a value specific for the given OPC server;
<*sub-condition*>	- sub-condition name – a value specific for the given OPC server;
<*asix alarm*>	- alarm number in the **asix** system;
<*Atribute ID*>	- an optional field that indicates attribute containing the variable the value of which will be read and sent to the asix system; the value may by displayed in the alarm text..

EXAMPLE

alarm/attribute/trip/start, My.Source, Trip, COS, Default 55
alarm/attribute/trip/impulse, My.Source, Trip, COS, Default 56
alarm/attribute/trip/durable, My.Source, Trip, COS, Default 57

To display a variable value in an alarm text it is necessary to use the %f symbol in the alarm definition in the place where the value has to be displayed.

EXAMPLE

17, AL, Boiler temperature. The cv attribute value equals to %f
101, AL, Tape transport Speer equals to %f meters per second

12.1.6. Configuring Network Operation

Alarm system of AS32 program has built-in mechanisms of network operation. They become active if during starting the application ASLINK module will also be started.

Alarm system may operate in one of four modes. This mode is set with use of ***Work mode of alarms system*** parameter, declared for application configuration file in Architect program > *Fields and Computers* > *Alarms System* module > *Alarms t*ab:

Operator - active mode, which is characterized by full possibilities of handling the active alarms. The active station may detect the alarms. An operator has a possibility to confirm alarms and change of disabling and filters. In network systems, operator's stations may be connected with of the group, which handle the same set of alarms. This is synchronous handling. Confirmation of an alarm, a change of disabling or a filter on one station is transmitted immediately to all stations of the group. During computer restart, supplementation of files of historical log from other working stations of a group takes place. Active stations may also make available their archives of alarms for Historical_viewer stations.

Historical viewer - passive mode, it is deprived of possibility of handling the active alarms. Strategies of alarm detection are not started up. Historical_server stations are used to overview historical logs, which are transferred from operator stations.

Local - this mode, from the point of view of local operation, gives the same possibility as Operator one. The only restriction is the lack of possibility of connection with other stations to exchange data of alarms.

Historical server - from point of view of an operator, operation in historical alarm server mode is the same as in passive viewer mode. The essence of server mode consists in its service for the other network stations Historical alarm server during the entire period of its operation receives copies of alarm logs from all Operator stations, which it will be able to detect. The collected archives may then be available for Historical_viewer stations. The alarm server may be applied in two cases:

- in large installations (with great number of Historical_viewer stations) it enables to support operator stations
- in installations composed of two separate networks;
  Computer, so called gateway, which works in server mode, may make alarms available

from operator stations of the first network to Historical_viewer stations in the second network.

The essential concept, whose understanding is necessary to obtain proper network configuration of alarm installations, is so called Set of Alarms. The below mentioned elements composed the definition of set of alarms:

- files of alarm definitions;
- files of group definitions;
- installed strategies of alarm recognition.

Set of alarm is defined by 8-character name, which is declared in application configuration file in Station name parameter with use of Architect program > *Fields and Computers* > *Network module* module > *Standard* tab. The operator stations cooperate only when the same name of alarm set is given. It is essential that system designer could provide the compatibility of all elements included in the set. In case of alarm and group definition file, compatibility of names of file is sufficient (the program transfers automatically the content of files in case of discrepancies, the date of file saving is checked). Whereas, the parameters of strategies of alarm recognition must be compatible. This results from the fact that within the range of operation group of operators' stations, only one computer recognizes actively alarms and transfers them to other computers. Because it is not determined which station is the active one, thus each of them must be able to undertake detection of all alarms.

Historical_viewer stations do not depend on network names. They may collect historical archives of any working operator's station. Together with content of archives, the files of alarm and group definition are also transferred. The transferred archives are saved in sub-directories of alarm directory, which are created individually for each station, from which the archives are transferred. Name of directory is created on the basis of network resource name of connected computer. In a directory, besides archives and definition files, *alarm.ini* file is generated, in which information on computer, from which the archives had been sent, is saved. After initial complete transfer of files from active station, an automatic, periodical test of updating the changes taking place in archive condition is continued. The control station may operate being entirely separated from operator stations, and is used to browse the alarms, which are located in previously created archive files.

12.1.7. Exclusions and Filters

FILTERS
EXCLUSIONS

Excluding allows to eliminate the alarms, which, for the sake of damage of devices or links, are present on the active alarm list despite that you know they are not active at a moment.

In usual mode of operation, exclusions and filters are realized by side of a computer. However, there exist possibility of co-operation with mechanism of disabling of a controller. AS32 program is limited than only to operator interface, which enables to set a bit map of excluded alarms in controller memory. This mode of operation is included when ***Exclude map*** parameter is declared for application configuration file with use of Architect program:

Architect program > *Fields and Computers* > *Alarms System* module > *Maps* tab

At the same time, internal inspection of disabling and filters is switched off.

Alarms system operating in the mode, in which alarms excluding is limited only to disabling the monitoring alarms on masks, is possible. All events are still saved in log files. When alarm excluding is switched off, it is possible to review all registered events.

This mode is enabled with the Disable display of excluded alarms parameter set up for application configuration file with use of:

Architect program > *Fields and Computers* > *Alarms System* module > *Advanced* tab

In some applications it is necessary to simultaneously disable a large number of alarms. It is connected with special modes of installation operation – startup phase, maintenance works, disconnecting a part of installation. Disabling by files mechanism can be then applied. This operation consists in creating the files containing alarm numbers. In this way predefined alarm sets can be then excluded with a single operation. There is possibility of writing the alarms exclude status to the file, and to perform disabling with use of the file contents.

All exclude files are stored in working directory of alarms system (ALARMS by default) and have EXL extension. These are files in text format that contain alarms numbers written in separate lines. Creation the exclude files is possible with help of a standard text editor working in ASCII mode (for example Notepad program of the Windows system). Using the exclude files is done in alarms exclusions window. It is possible to control exclusions by means of EXCLUDE operator action.

Alarms exclusions window is open by the alarm window toolbar. The window looks like as follows:

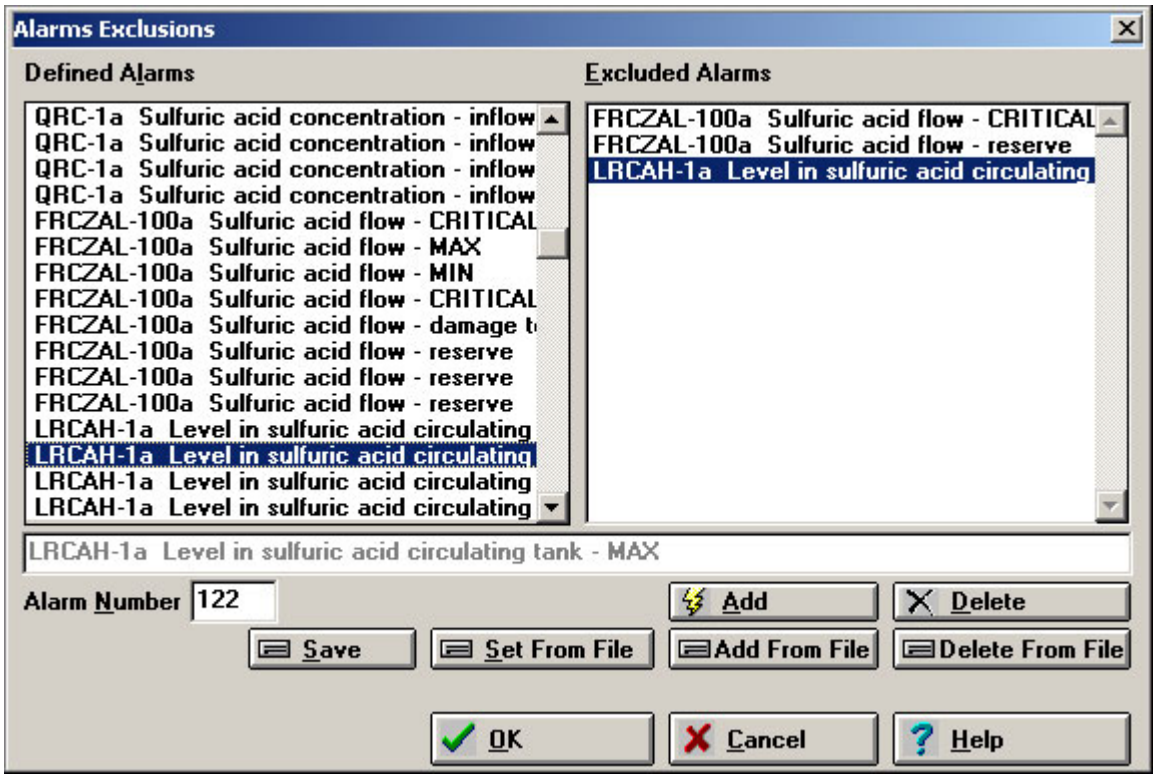


Figure 12-1. 'Alarms Exclusions' Window.

The exclude files are handled with *Save, Set From File, Add From File* and *Delete From File* buttons:

Save - button causes writing to the file numbers of all alarms present on the *Excluded Alarms* list in the moment of clicking on the button.

Set From File - button triggers reading the selected file and copying to the *Excluded Alarms* list alarms with numbers saved in the selected file. At the same time all alarms with numbers not present in the file are removed from *Excluded Alarms* list. New disabling status will be identical to the status written in the file.

Add From File - button causes reading the selected file and adding alarms with numbers written in the selected file to the *Excluded Alarms* list.

Delete From File - button triggers reading selected file and removing the alarms with numbers written in the selected file from the *Excluded Alarms* list.

Set From File, Add From File and *Delete From File* operations don't perform any changes in disabling status. Only clicking the *OK* button triggers execution of new settings. It also causes writing to the alarms log the information about any changes in alarms exclude states.

All above operations cause the window shown below to be displayed:

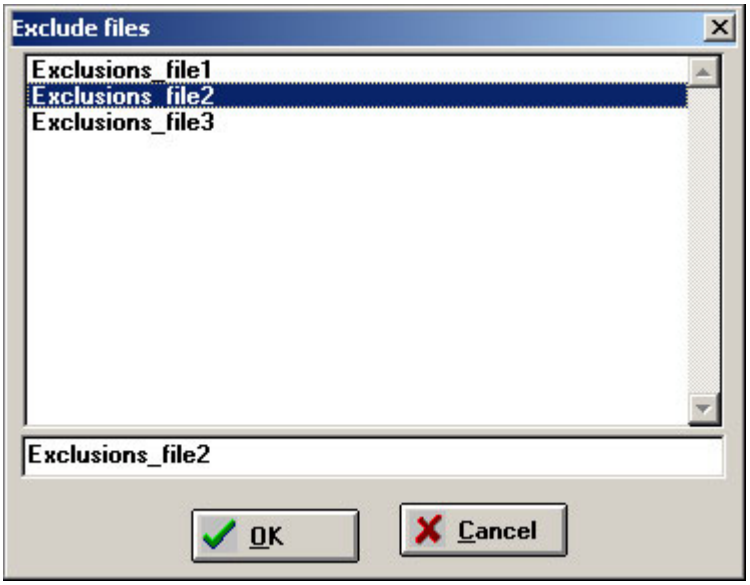


Figure 12-2. 'Exclude files' Window.

On the list names of the files of all previously saved exclude sets are displayed. It is recommended to give the files descriptive names, which allows easy identification of each sets' purpose.

Operator may choose the set name from the list or enter it manually in the edition field.

REMARK *All exclude files are stored in the working directory of alarms system (ALARMS by default) and have EXL extension. These are files in text format– contain alarm numbers written in separate lines. Creating the disabling files is possible with help of a standard text editor working in ASCII mode (for example Notepad program of the Windows system). In network systems the exclude files are not synchronized automatically between linked stations*.

ADVICE – How to Quickly Remove All Exclusions

The easiest way to remove all exclusions is using an empty exclude set file. To do this, at first the exclude file should be saved in the moment, when *Excluded Alarms* list is empty and give it a name, for example. „No Exclude". From this moment, the operator disposes the set, which, used in *Set from file* operation, will cause removing entire contents of *Excluded Alarms* list.

Filters

Periodic filtering eliminates the alarms, which appear for short time (they flicker) filling alarm archives with large quantity of events. After the filter is set, an alarm is reported only when it lasts sufficiently long time.

The **asix** system has two algorithms of alarm filtering. Selection of the algorithm is made by appropriate declaration in Filter method parameter set up with use of Architect:

Architect program > *Fields and Computers* > *Alarms System* module > *Advanced* tab:

- ***Filtration of short alarms appearing for a time shorter than declared***;
- ***Filtration of fast sentences: alarm appearance-disappearance-alarm appearance***.

Algorithm No. 1

Filtering time t_f - is turned-on by the raising and falling edge of signal
- is set individually for each signal

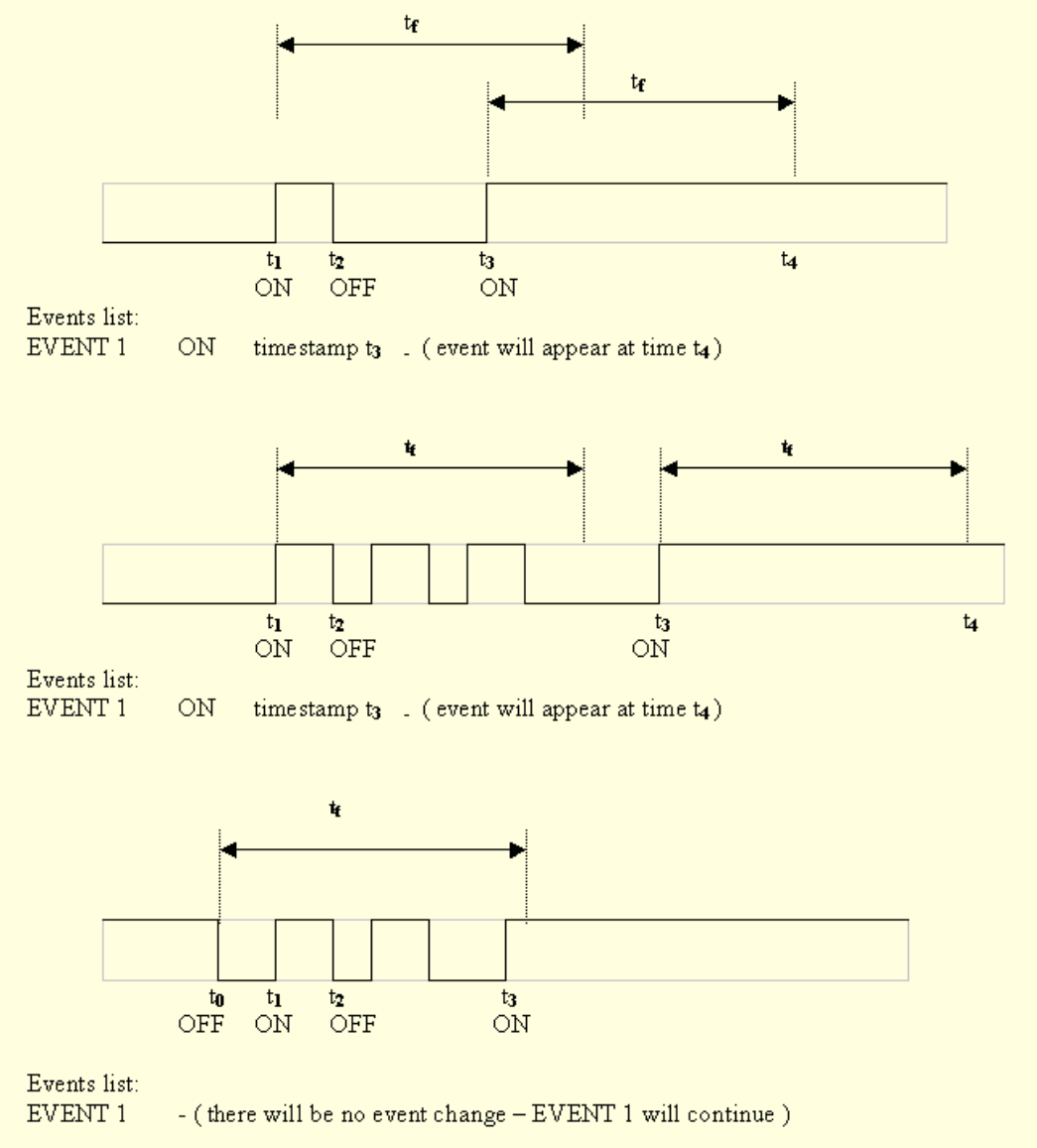


Figure 12-3. The Algorithm of Alarm Filtering (1).

Algorithm No. 2

Filtering time t_f - is turned-on only on the raising edge of signal
- is set individually for each signal

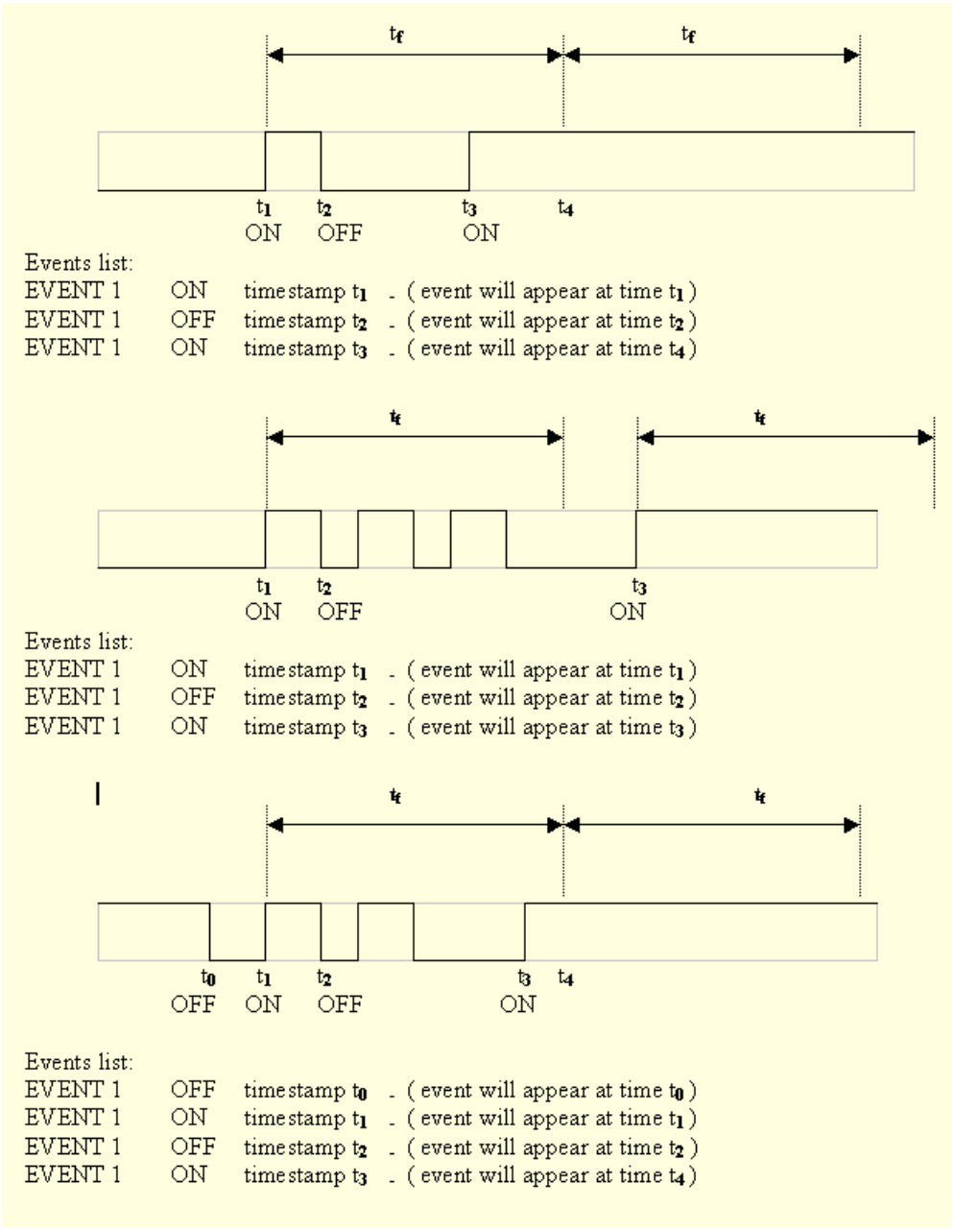


Figure 12-4. The Algorithm of Alarm Filtering (2).

12.1.8. Sound Signals

AS32 program is provided with mechanisms of sound signalling the alarm occurrence. By default, in case of events of type: warning, alarm and critical alarm the simple sound signals is generated, which do not require that any equipment is to be installed. If, however, a computer is provided with sound card, you may declare WAV type files, which shall be played while an event occurs.

Alarm signals are declared for application configuration file with use of:

Architect program > *Fields and Computers* > *Alarms System* > *Signalling* tab > parameters: ***Alarm signal***, ***Warning signal***, ***Important signal***.

Usually occurrence of a signal causes single sound signal to be generated. Using a ***Long alarm*** (Architect program > *Fields and Computers* > *Alarms System* > *Signalling* tab) parameter you declare that signal will be repeated until it is switched-off with a CLEAR_SOUND action.

If sound signal is performed by a controller, then individual signal for each alarm is possible. Co-operation with a controller is based on existence of bit map of signals (by default, bells and buzzers), which are read and stored by means of AS program. Existence of signal maps may be declared in ***Buzzers map*** and ***Exclude map*** parameters (Architect program > *Fields and Computers* > *Alarms System* > *Maps* tab). Using these items causes that the command opening the signal handling window appears on alarm mask.

With the ***Long alarm*** parameter active a priority sound alarm signalling is working where sound signal of alarm with higher priority is not disabled by the alarm signal with lower priority. For example, the signal of critical alarm signal is not disabled by the warning signal.

12.1.9. Control Log

The mechanism of control log allows recording in the alarm log the selected control operations performed by the operator. A principle of log functioning consists in binding variable names with alarm numbers. Performing successful variable control, which has an alarm assigned, causes generating the alarm. In the alarm text the sent control value may be placed.

Configuring the control log consists of three parts. First one or more files containing variable names bindings with alarm numbers should be prepared. These are text files, where every line gives one binding in a following form:

<*variable name*> , <*alarm number*>

Following step is preparing the texts of alarms that should be displayed in the moment of performing the control. In the alarm text, formatting characters can be placed (like %d, %f, %4.1f), and their usage will cause inserting into the content of the alarm the value of controlled variable.

In case of using the network mode of alarm system operation, all controls log alarms should be defined as system ones. It will allow correct broadcasting alarms in the stations network.

It is also necessary to place a **CONTROLS_LOG** entry, which turns support for the controls log on. This entry is declared for application configuration file with use of:

Architect program > *Fields and Computers* > *Miscellaneous* module > *Directly entered options*:

Section name: ALARMS_SYSTEM
Option name: CONTROLS_LOG
Option value: *junction_file_1 [,junction_file_2,..]*

Meaning	- item declaring names of files where the associations between variables names and numbers of alarms are defined. An item usage causes automatic switching-on of controls log procedures.
Default value	- controls log is not used.
Defining	- manually

AsAudit module ensures more advanced registration of controls.

See: *AsAudit – User's Manual*.

12.1.10. Operator Actions

CLEAR_SOUND
CONNECT_ALARMS
ACKNOWLEDGE_ALARMS

See: 16.2. Description of Actions.

12.2. Alarm Masks

In **asix** system, two types of alarm masks may be distinguished:

active alarm masks	- masks showing the list of active alarms, i.e. alarms, which had been detected but were not completed yet.
historical alarm masks	- masks showing the full history of alarms conditions change. Both occurrences as well as decays of alarms are recorded. The history saved on disk may include a lot of days. In case of systems operating in historical viewer mode, alarm archives may come from other stations.

Besides possibility of browsing, the alarm masks create the possibility to execute a series of managing operations such as:

- confirmation of alarms (only on active alarm masks),
- selection of presented alarms,
- excluding selected alarms from monitoring
- installing time filters for moments of occurrence and decays of alarms,
- printing of alarms for pre-set period of time,
- transition to schematic mask associated with given alarm,
- switching over to other stations of historical archives,
- control of sound signalling.

Beside of some specific features resulting from assignment of described features, alarm masks have all features of schematic masks, i.e. they may be moved, minimized to an icon, maximized, etc. Handling of these standard operations is the same as in case of schematic mask. The further part of the section relates essentially to elements, which are typical for alarm masks only.

12.2.1. Handling of Alarm Masks

Acknowledgement
Criteria
Disablings
Filters
Signals
Printout
Mask
Definition
Connect

Alarm masks are open using the same method as in case of synoptic masks: by means of pressing the proper keyboard shortcuts or clicking with the mouse at suitable place of synoptic mask. Besides there is possibility to open alarm masks (in their standard form) by means of execution of commands TOOLS.ACTIVE_ALARMS_WINDOW or TOOLS.HISTORICAL_ALARMS_WINDOW in Control Panel window.

Masks of active and historical alarms may take different form depending on their parameters set by application designer. They may be displayed as large window providing all control elements, they may also, in extreme form, be composed of one alarm line without any control elements.

Below, the entire active alarm mask is shown opened with a command of Control Panel.

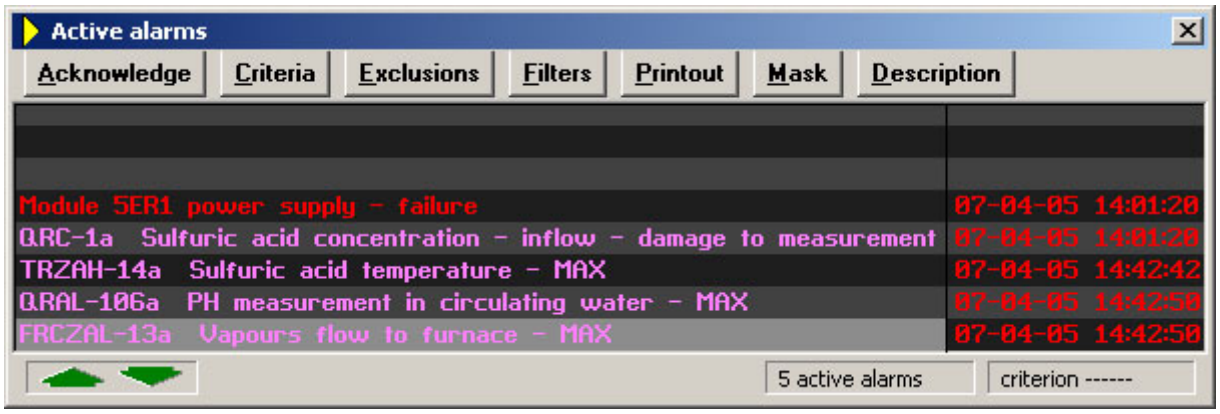


Figure 12-5. 'Active Alarms' Window.

The full version of active alarm mask is composed of the following elements:

- frame;
- heading line with keys of system menu, transition into an icon and to the full screen;
- tool bar used for management execution;
- alarm list, of which each line is composed of alarm text as well as the time of it occurrence;
- status line composed of arrows used to move along the alarm list as well as a field showing number of active alarms.

The complete version of historical alarm mask is presented below:

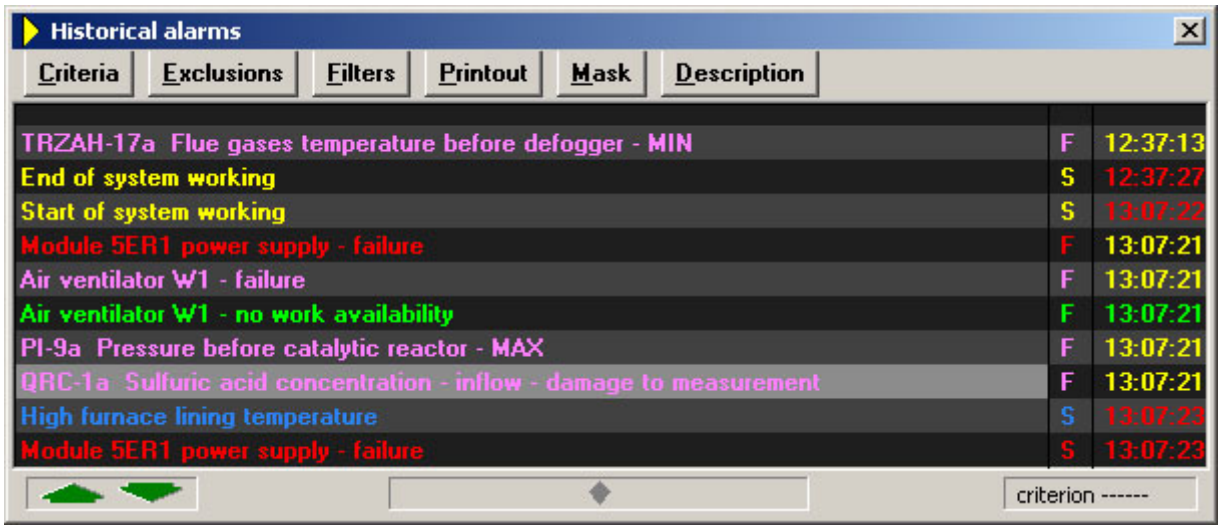


Figure 12-6. 'Historical Alarms' Window.

In this version, historical alarm mask is composed of the following elements:

- frame;
- heading line with keys of system menu, change into an icon and to the full screen;
- tool bar used for execution of managing operations;
- alarms area, of which each line is composed of alarm text, identifier of condition change and time of event occurrence;
- status line composed of arrows used to move along the alarm list, a field showing the range of time limitation of displayed alarms as well as a field that gives the set of used selection criteria.

Alarm texts are displayed in colors, which depend on alarm type, which they describe. There is the following assignment of text colors to alarm types, which are distinguished in **asix** system:

yellow color - system alarm, not connected with inspected process but related to internal system operation events,
green color - message,
blue color - warning,
purple color - normal alarm,
red color - important alarm.

Additionally, an alarm being in selection condition is displayed on blue background.

NOTE *In case of active alarm mask, such parameters setting is possible that all alarms, regardless of their types, are displayed in white color on red background. This enables to create so called the recent alarm masks (showing only the recent active alarm), on which occurrence of alarm causes displaying it in red contrast color. Occurrence of alarm of warning type, usual alarm and important alarm is additionally signaled with sound signalling (which depend on alarm type)*

Range of colors of displayed event times is as follows:

light-red color - active alarm, unconfirmed,
red color - active alarm, confirmed,
green color - completed alarm, confirmed,
yellow color - completed alarm, unconfirmed.

Identifiers of condition alteration displayed on historical alarm mask determine type of event, which occurred for case of presented alarm. The below mentioned values of identifiers are used:

S - start of alarm,
F - end (finish) of alarm,
E - forced completion of alarm caused by excluding of alarm from monitoring by an operator.

Colors used on alarm masks may be changed by means of items of section ALARM_COLORS.
Alarm masks show once the limited numbers of alarm events (usually the recent ones) An operator has possibility to change the range of shown alarms using keys *PgUp*, *PgDn*, *Home*, *End* as well as with arrow keys. If alarm mask has status line it is possible to move among alarms by means of a mouse. Clicking with left mouse key on images of arrows in status line will cause the change of selected alarms by one position, clicking with right mouse key will

cause the change by the whole page. The above operations allow also changing the selected alarm.

Presence of tool bar on alarm mask enables to perform managing operations of the alarm system. Execution of operation is initiated by clicking the mouse on selected button or depressing the combination *Alt-character* on the keyboard, where used *character* corresponds to the letter distinguished on description of a button. Meaning of separate operations is described below:

Acknowledge - execution of operation of alarm confirmation. All displayed on a mask alarms are confirmed. However two additional conditions are to be met: an alarm should not be completed and it must be selected, if it is an "important" type alarm. The alarms can be confirmed selectively by double-clicking the mouse on the alarm's line.

Criteria - determination of conditions, which are to be met to display an alarm on the mask. There is possibility to use 6 criteria of selection:

- time of event occurrence (only for historical alarms),
- alarm describing text ,
- alarm type,
- alarm condition,
- assignment of alarm to a group,
- alarm number.

It is possible to select an arbitrary set of criteria, which are to be used as selection conditions. Special attention is to be paid on time period criterion. It is useful, first of all, at browsing of alarms, which have had occurred in distant past. It allows find quickly an interesting part of alarm history. The second advantage of using this criterion is limitation of alarm searching, in case of using criteria, which are met by small number of alarm, and archive collected includes data from many days. Limiting the period of time allows significantly increase system response speed.

The selection criteria can be saved to the file and the criteria can be set accordingly to the file's contents from the historical alarms masks alarms selection window. The criteria files are stored in alarm archive directory and have SEL extension.

Below, selection window of historical alarm is shown:

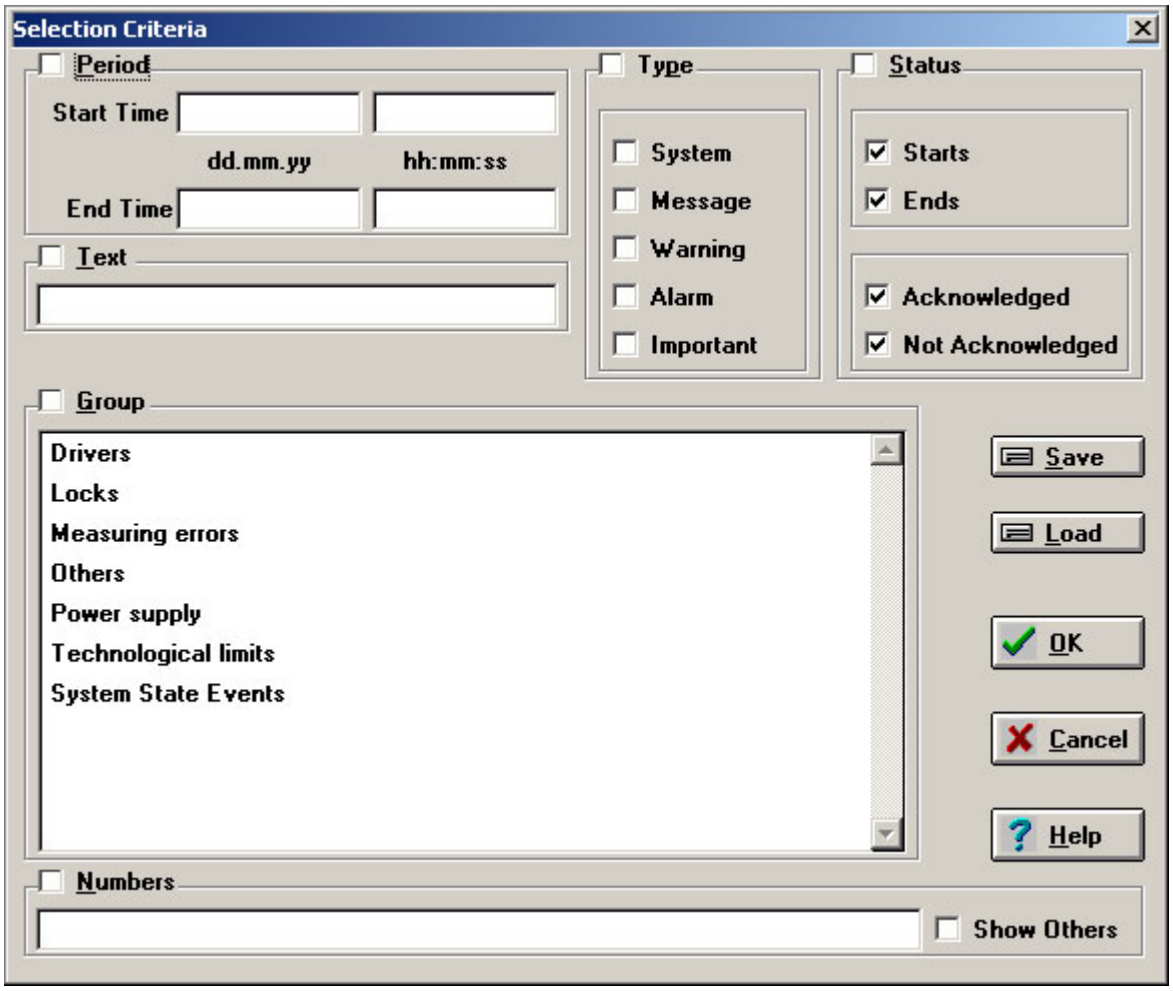


Figure 12-7.'Selection Criteria' Window.

Selection window is composed of six groups corresponding to separate criteria**.** It is necessary to set the selection check box, located at the beginning of the criteria section, if the criterion has to be taken into consideration.

In case of usage time criterion, leaving an empty field *Start_time* denotes using the oldest stored alarms, and an empty field *End Time* denotes displaying new-coming alarms. Text criterion is defined by entering a string of text to *Text* field, which should appear in the text of alarm message. Special characters may appear in the text pattern, i.e. wildcards * and ?. Character * denotes, that in its place in alarm description, any quantity of identified characters may appear. Character ? denotes that any single character may occur in its place. The most typical declaration of the pattern is as follows:

text - selects alarms, description of which starts from "text" string;

*text - selects alarms with "text" string appearing in any place.

Group criterion is declared by selection the group on the name list, to which the displayed alarms should belong. You have possibility to select up to 10 names of groups at the same time.

The number criterion allows limiting the list of displayed alarms to one defined by numbers. A comma-delimited number list can be declared (for example

2,34,789), the range of alarm numbers (for example 3-128,300-572) or both methods of specifying the alarms to be displayed can be joined (for example 2,4,5-89). Selection field *Show another* is used to indicate, that alarms determined by list of numbers should be deleted from the displayed list (negative selection).

Save and *Read* buttons are used for saving the settings of selection criteria to the disk and for reading previously saved files. Those files are located in alarm archives directory and have *.SEL extension. Pressing one of these buttons triggers opening the window shown below:

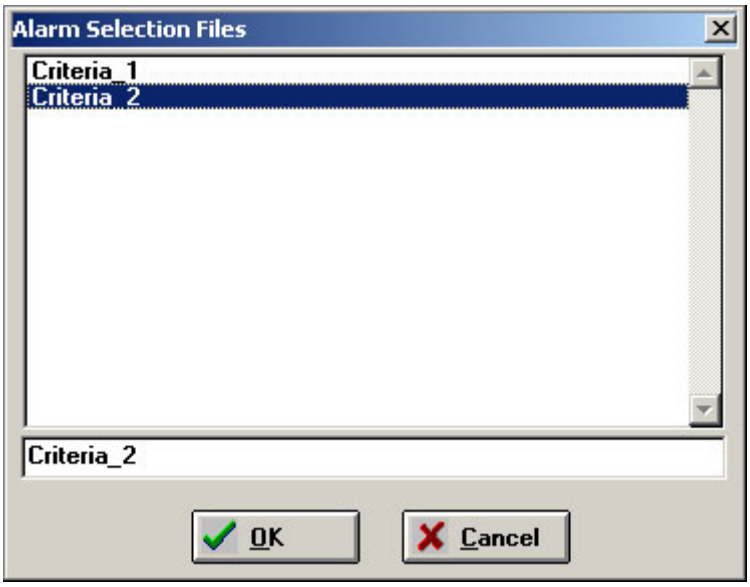


Figure 12-8. 'Alarm Selection Files' Window.

In the upper section of the window appears a file list with saved selection criteria, which are located in alarm archive directory. In the lower edit field the name of the file to be saved can be entered.

Exclusions - this command allows disabling the monitoring of some alarms by **asix** system. It is used for alarms, which are still in active state because of device failure or measurement interfaces and you know, that they are not really pending. Alarm disabling enables deleting such alarms of active alarm list. The window used for disabling the alarms is presented below:

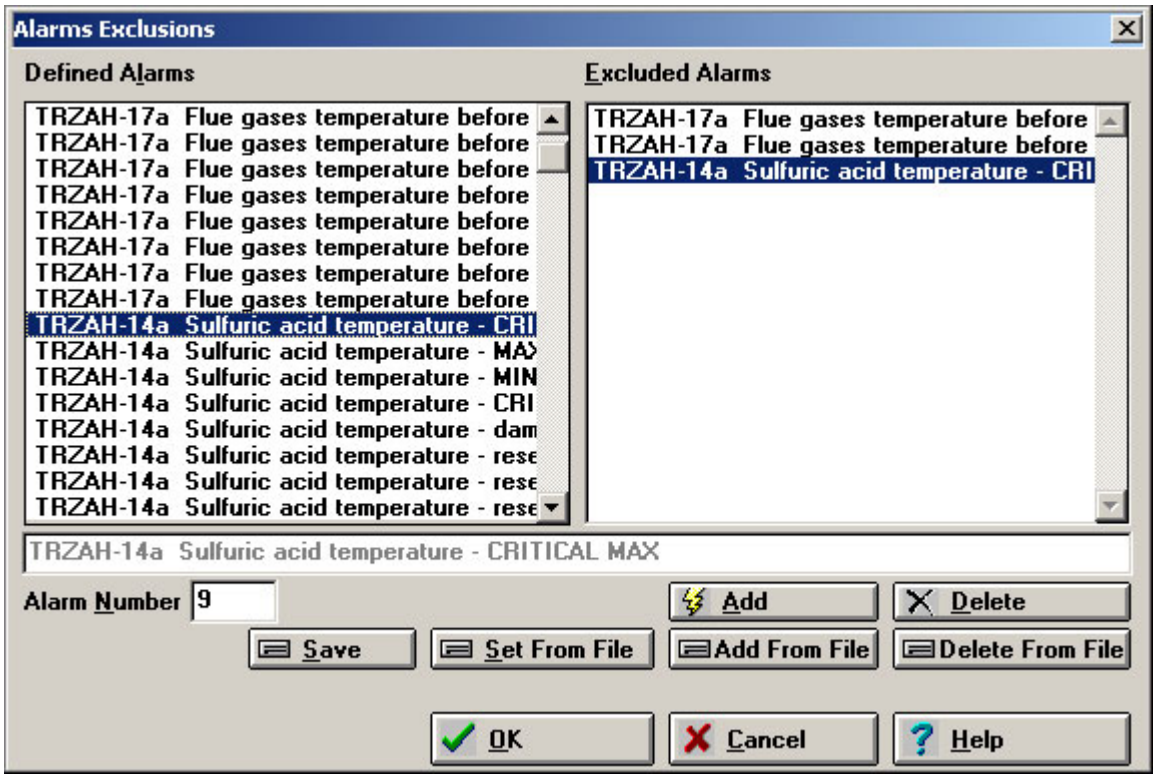


Figure 12-9. 'Alarms Exclusions' Window.

The disabling window is composed of two lists: in the first one all alarms defined in the system are shown, in the second one - the alarms, which are disabled at present. Disabling an alarm consists in i selection of defined alarm list and depressing the key *Add*. Restart of alarm condition inspection consists in selection of an alarm on disabled alarm list and depressing the key *Delete*. Due to the fact that number of defined alarms could be very long, special convenience has been used to select alarms of this list. In the moment when disabling window appears, an automatic selection of the currently pointed (on the active alarms list) alarm follows. Besides, there is a possibility of selection by means of writing the text of alarm, which is to be disabled. The text being entered does not appear on the screen but prompt selection of alarms is executed, the description beginning of which is the same as given text. Depressing *Enter* key after the text had been entered causes consecutive transition to alarms, which begin with proper text.

Each disabling and restart of inspection of an alarm is written through generating the system alarm, which describes the action executed by an operator.

It is possible to save the alarm exclude status to the file, and also to perform the disabling according to the file's contents. Exclude files are stored in the alarm archive directory and have the extension EXI. When writing into the file the only thing you should pass is the file name, the extension and location will be declared automatically. The buttons: *Save, Set From File, Add From File* and *Delete From File* open the window *'Exclude Files'*.

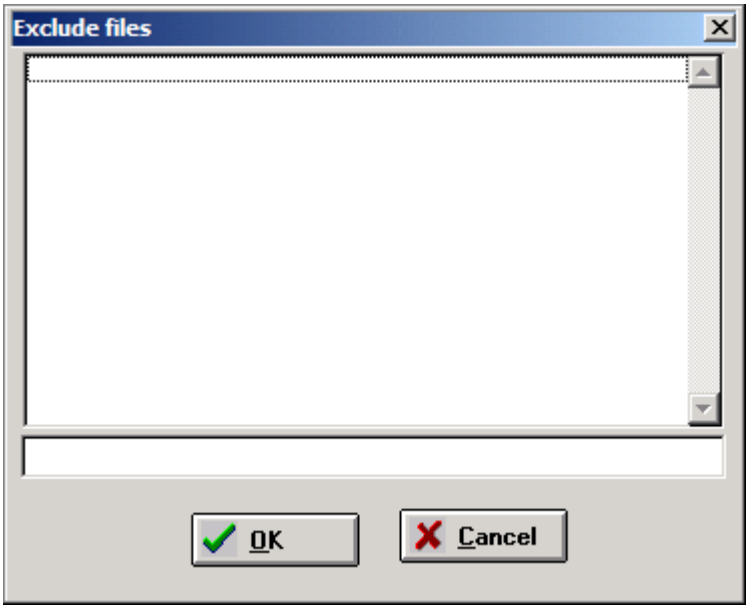


Figure 12-10. 'Exclude Files' Window.

Filters - this command enables to enter a time filter to detect changes of condition of some alarms. It is used for these alarms, which tend to so called "flickering", that makes unnecessary overloading of alarm masks with permanently repeated events. It is possible to declare one of two methods of filtering the alarms in the application's initialization file. First filtering method causes, that only status changes lasting longer than a defined minimum are recorded. The second method eliminates quick appearing/fading/appearing sequences of alarms.

A window used for enabling alarm filtering is shown below:

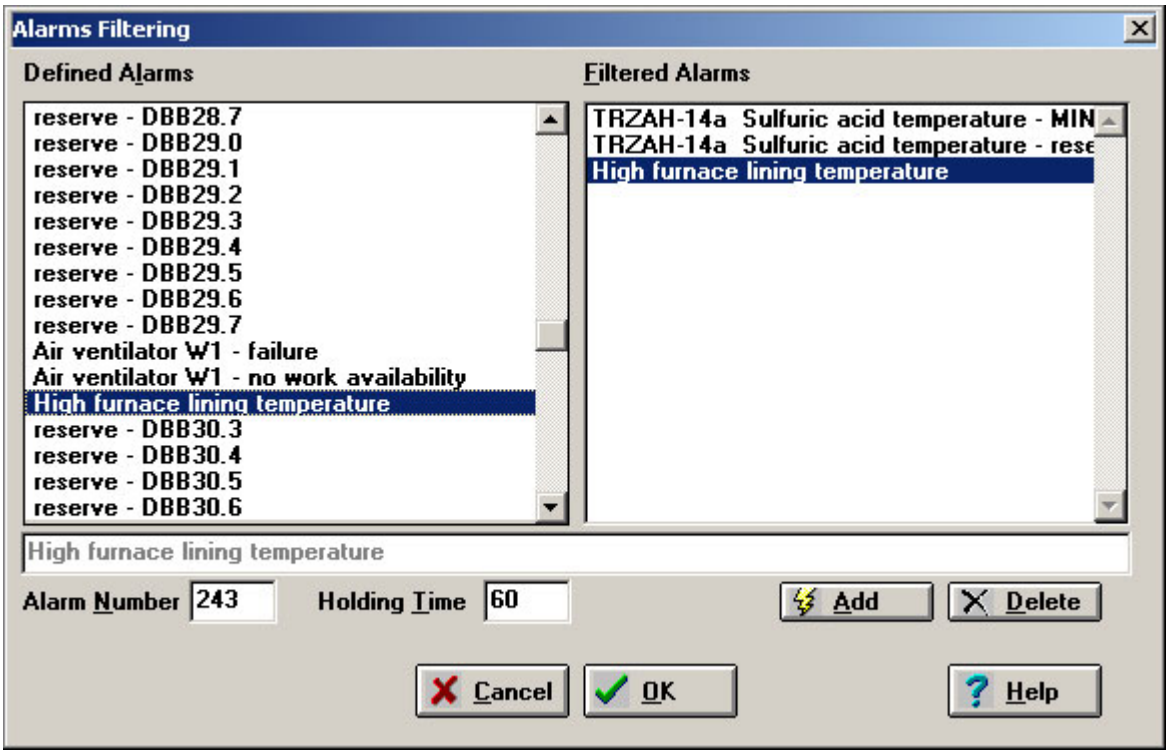


Figure 12-11. 'Alarms Filtering' Window.

Window of filtering is composed of two groups: in one of them all alarms defined in the system are shown, in the second one the alarms, which are filtered at present. Switching on the alarm filtering consists in selection the alarm on defined alarm list and depressing the key *Add*. Restarting the usual handling of alarm consists in selection of alarm on list of filtered alarms and depressing the *Delete* key. Each of selected alarms has individually defined time of filtering. It appears in edition field below the list of filtered alarms at the moment of selection of an alarm. This field is used to change the filtering time. Filtering time is expressed is seconds.

Because the list of defined alarms may be very long, special conveniences had been introduced provided for selection of alarms in this list. While filtering window appears, an automatic selection of the currently pointed (on the alarm mask) alarm follows. Moreover, there is possibility of selection through writing the alarm text, which is to be filtered. Text being filtered does not appear on the screen but prompt selection of alarms is executed, the description beginning of which is the same as given text. Depressing the Enter key after the text is entered causes consecutive transition to alarm, which begin with proper text.

Each switching on the filtering and restart of usual handling of an alarm is recorded by generating the system alarm describing the action made by the operator.

Signals - a command, which opens dialog window used for setting the parameters of sound signalling of alarms, executed by PLC controller.

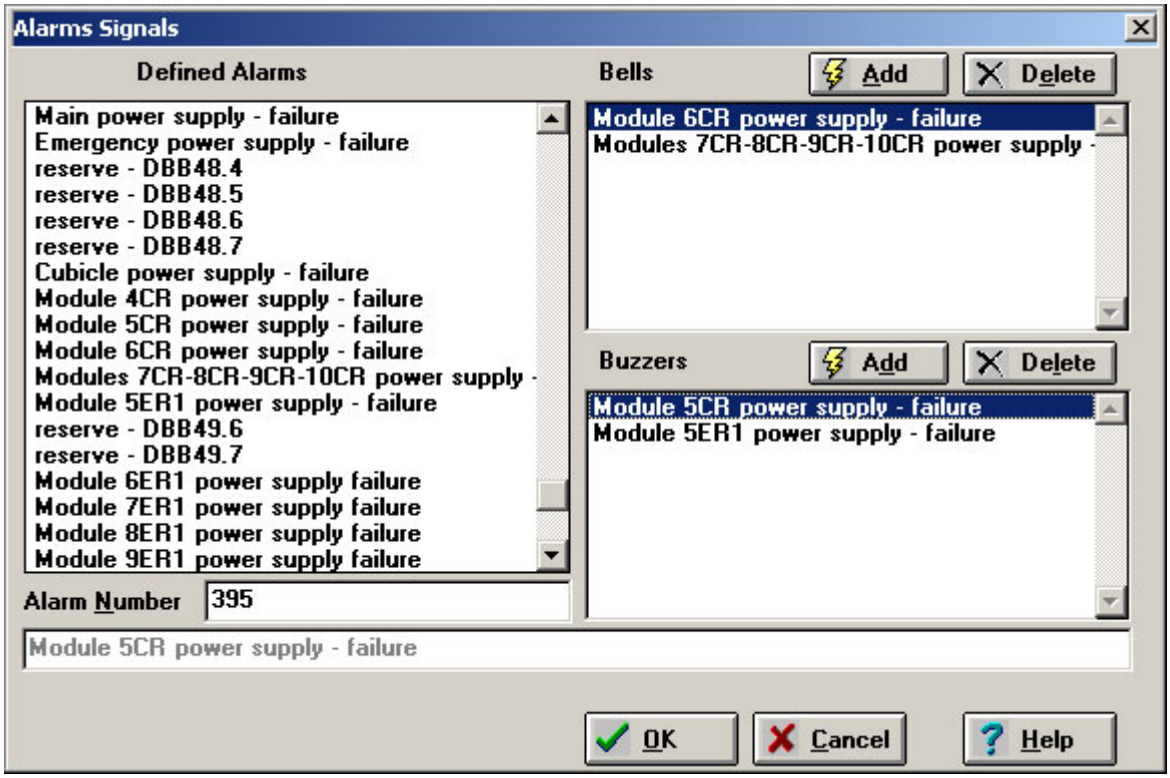


Figure 12-12. 'Alarms Signals' Window.

On the left side of the window there is a list of all defined alarms. On the right side of the window there are two lists, in which alarms with assigned bell and

buzzer signalling are shown. Switching on signal for an alarm consists in its selection of defined alarms list and depressing a suitable key *Add*. To delete (erase) the sound signalling, you should to point the alarm on list of bells and buzzers and then to depress a proper key *Delete*. Depressing the key *OK* will cause creation of bit maps of signals and transmission to the controller.

Printout - this command allows to print the alarms, which changed their state in selected period of time. Only these alarms are printed, which meet selection criteria set at present (except for time criterion, because the scope of time is defined here separately). The alarm printout window is shown below:

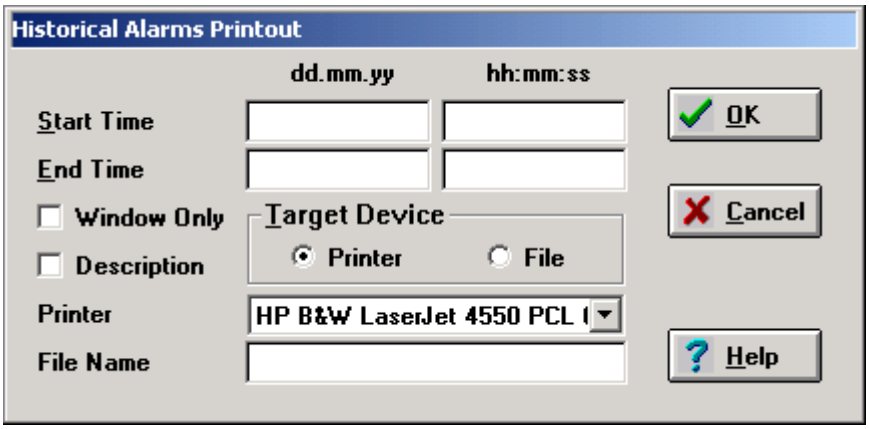


Figure 12-13. 'Historical Alarms Printout' Window.

Leaving an empty field *Start Time* denotes execution of a printout of the oldest saved alarms, and an empty field *End Time* denotes printing until the newest alarm. Information, which is to be printed may be saved in a file (by means of selection of an option *File* and setting the field *Filename*, where from it may be printed later, or it may serve as a basis of more detailed analyses. The *Window Only* selection field allows to print historical alarms currently visible on the alarms mask, chosen previously according to specified set of criteria, and *Description* selection field allows assigning the printout an additional information about the source of alarms, period they are printed for and used criteria.

Mask - execution of this command causes transition to mask associated with an alarm, selected at present on alarm mask. Interconnection of alarms with proper schematic masks is made by system designer.

Definition - this command has only auxiliary meaning. It enables checking the whole definition of selected alarm on the mask. Executions of this command will causes to open the below window.

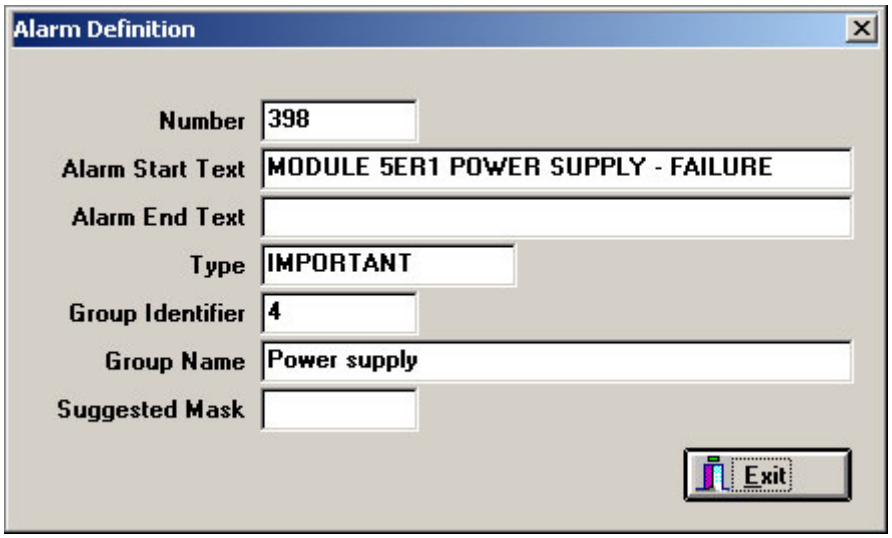


Figure 12-14. 'Alarm Definition' Window.

Entering in the Number field alarm number different than just selected will cause displaying its definition (even if given alarm doesn't appear on list displayed on the alarm mask)

Connect - this command is available only on historical alarm masks of historical viewer systems. It enables to select the source of alarm origin. Execution of the command will display the below window:

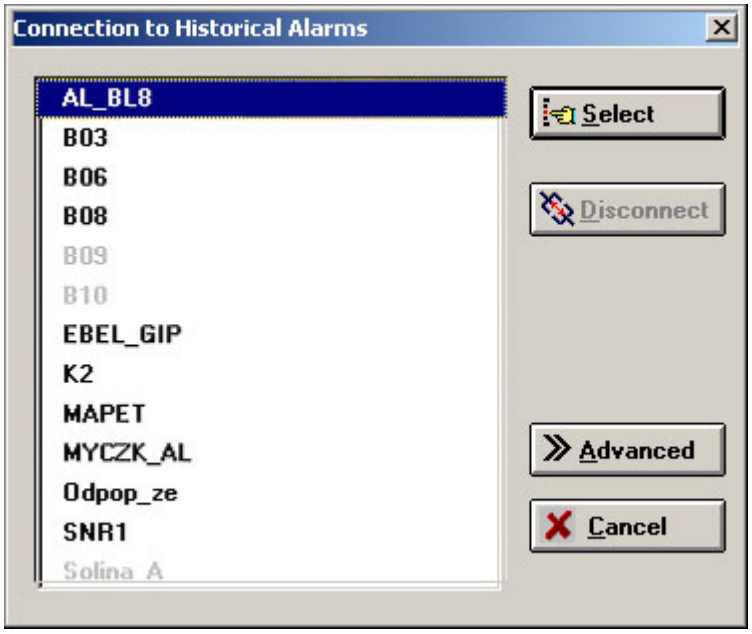


Figure 12-15. 'Connection to Historical Alarms' Window.

It shows the list of available resources – black color, and the list of resources, which are on station's local disk and were downloaded in result of previous connections with alarm servers– gray color. Choice of the resource to be connected is made by the *Select* button.

Advanced button opens the following window:

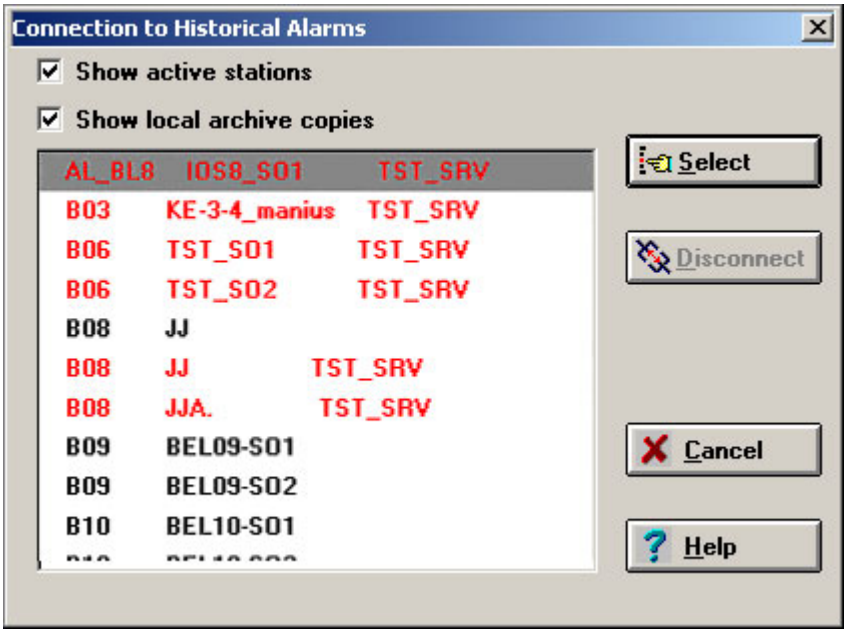


Figure 12-16. 'Connection to Historical Alarms' Window - Advanced Option.

The basic part of the window is list of stations, to which the operator may make a connection. Description of the station consists of the network name of the source as well as name of a computer. The third name may also appear. This is the name of a server of historical alarms via which the alarms are transmitted. Items are displayed in two colors. Red color denotes connection with the station operating at present. In case of selection of such connection updating of archive and files shall take place from operating operator station. Also during displaying a mask a trial shall be made (using 1 minute cycle) to transmit changes in archive files. An item in black color denotes connecting up to a directory, in which the achieves have been saved earlier (with red connection or manually by means of a diskette). In this case a trial of archive updating will not be made, a presence of operator's station is not required. Selection fields *Show active stations* and *Show local archive copies* allow shortening the set of items in connection list. Each alarm mask may be connected to another alarm source.

Some management operations may be executed regardless of presence of key bar. It enables the basic operations to be executed on simplified alarm masks (for instance a mask showing only the last active alarm) Keyboard keys, which have special meaning on alarm masks are:

Enter - confirmation of the alarm,
Double click - confirmation of the alarm,
Grey + - transition to associated mask.

The alarm masks has context menu opened with right mouse button. In this menu the number of selected alarm and following commands are shown:

- passage to the mask associated with the alarm(if such mask has been defined),
- definition of the lower limit of alarm printouts(based on the time of selected alarm),

- definition of the upper limit of alarm printouts(based on the time of selected alarm).

12.2.2. Defining Alarm Masks

The method of defining alarm masks is similar to this one, in which common schematic masks are created. Creation new active alarm mask or historical mask requires executing one of the commands:

MASKS.NEW_MASK.ACTIVE _ALARMS or
MASKS.NEW_MASK.HISTORICAL_ALARMS.

Mask defining window will be open. Below the mask defining window of historical alarms is shown.

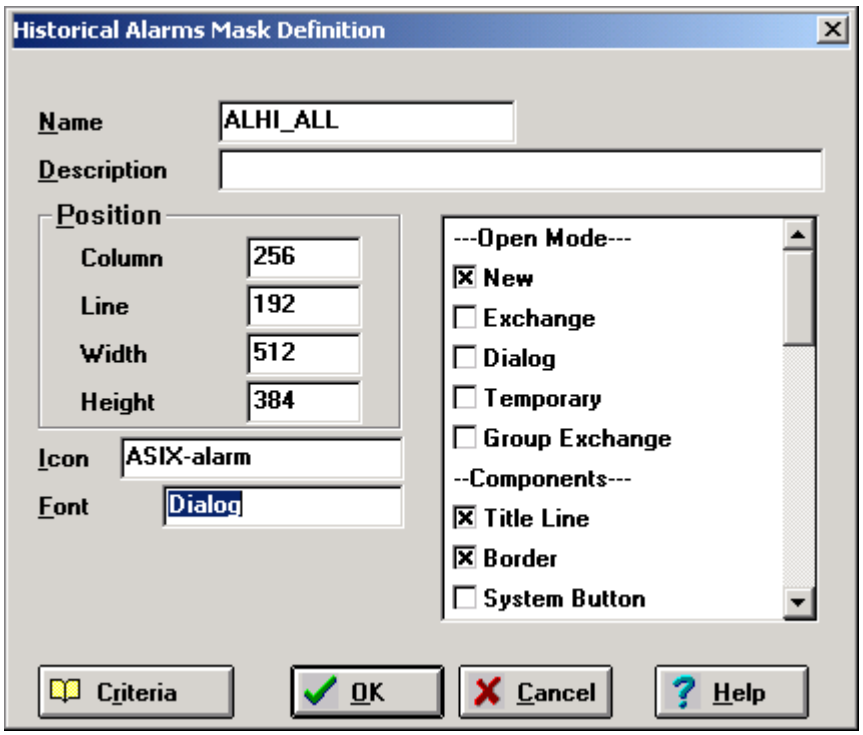


Figure 12-17. 'Historical Alarms Mask Definition' Window.

As you can see, alarm mask defining window contains mostly the same fields as in case of schematic masks defining window. Managing and meaning of these fields is also the same. The differences are as follows:

Check box *Toolbar* - a field in the mask attribute list. Checking of this field is necessary to add tool bar, used to execute alarm managing operation, to alarm mask.

Check box *Status line* - a field of mask attribute list. Checking of this field is necessary to add status line to alarm mask

Check box *No Events Column* - selecting this field causes, that the event type (beginning/end of the event) is not

	registered on the historical alarms list. The only method of distinguishing the beginnings from endings of the alarms is to define suitable alarm texts for the beginning and ending of the alarm's activity.
Check box *No Date*	- the date of the event is not displayed on the historical alarms list. Setting has sense, when the alarms list is limited to one day.
Check box *Red Mode*	- setting the field will cause that all alarms are displayed white on red background. The field is used only for active alarm masks.
Check box *Frame Select*	- changes the way of pointing the current position out on the mask. Instead of changing the background color, the alarm is indicated by drawing a border around the alarm's text.
Check box *Date on Toolbar*	- causes displaying a field on the mask's button bar showing the date of alarm, located on the last lower position of the alarms list– regards only the historical alarms mask.
Check box *Date Change Lines*	- causes inserting the line on the mask, informing about the change of date between each pair of displayed events differing in date. Displaying the date change line is not dependant on the selection criteria– it only concerns the historical alarms mask
section *Toolbar Buttons*	- with the ***Toolbar Buttons*** section it can be chosen, which buttons of the toolbar should appear on the alarm mask.
Button *Criteria*	- depressing this button will cause opening the alarm selection window, in which the origin criteria set for created mask may be defined. The selection window being open differs from selection window used in the moment when application operates in this fact that there is no possibility to set the time and number criteria.

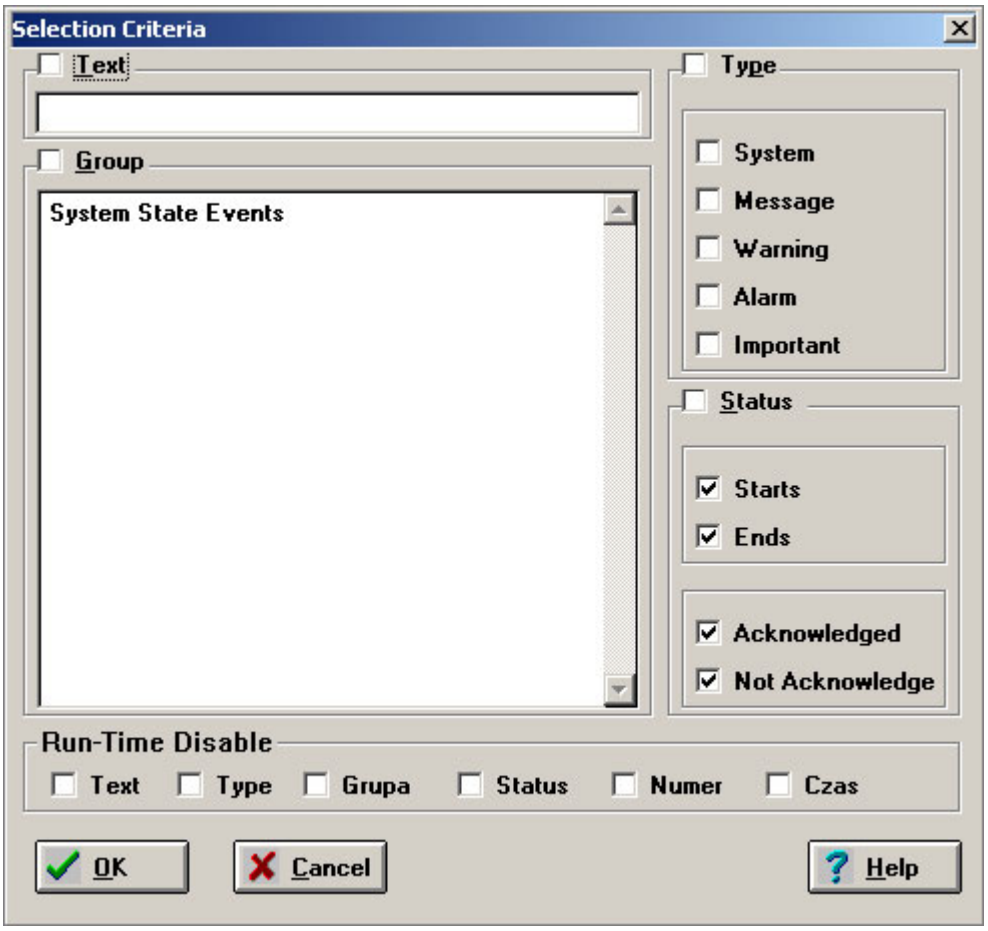


Figure 12-18. 'Selection Criteria' Window.

The alarm mask, which selection criteria were set during designing process, will be open with predefined criteria active. The group of selection fields „Run-Time Disable" is used for choosing the set of criteria, that will be unchangeable for the operator in the time of the application run. The status of other criteria can be influenced by the operator from the normal criteria selection window.

Edition field *Font* - in this field you specify the name of font, in which description of alarms are to be displayed.

Check box *Special events* - the field characteristic for historical alarm masks; checking the box, causes the alarms recognized as "special" to be displayed with the '*' character.

There is the possibility of declaring the alarms being recognized as "special" when using the external alarm detection strategy, improved by the user of the **asix** system (more detailed information – technical support of the ASKOM company).

Size and layout of alarm mask may be given directly in defining window or it may be set by means of mouse. On alarm mask none visualization objects may be placed.

Definitions of alarm masks are stored in files of extension MSK (the same as schematic masks).

The method of using the alarm masks in an application is the same as for schematic ones. Opening and closing operations are executed by means of the same actions.

13. Report Generation Module

Generation of reports is the inherent feature of the **asix** system. Definitions of reports are created by means of the built-in language **ASTER** designed to creation of reports. This language will be described later.

The reports can be divided into the two groups:

- the reports displayed on synoptic diagrams performed by objects of the REPORT type, in further they will be called object reports;
- the reports that are stored on the disk and, if necessary, displayed in the windows; they are performed by a special service module that is to be run by means of menu or appropriate actions, in further they will be called disk reports.

13.1. Handling Reports

Definition Group
Function Group
Report Group

For the first case, definition of the report is considered as a fragment of object definition and is stored in the file containing the description of diagrams (**.msk*). The report, after having been calculated, is displayed on the synoptic diagrams only. The object report can be calculated either in the on-line, or on the operator's request, or periodically at the established time of day. The mode of report calculation is specified while setting parameters of the object. The object reports should be designed in such a manner that they will not contain too much information. This limit is constrained by small room for displaying reports of that type (the windows of synoptic diagrams cannot be scrolled and they must share screen area with other objects). The time horizon of such reports usually does not exceed 24 hours. More information about those reports you can find in the chapter that is devoted to REPORT class objects.

Definitions of disk reports are stored as separate files (**.r*). The reports can be calculated at a request or calculation may be executed automatically. The result reports are stored on disk as text files. The command TOOLS.REPORTS WINDOW in the Control Panel menu or Designer window is to be executed. Then the report window shown on the picture above will be presented on the screen. Displaying of the report window can also be carried out by means of executing the MAKE_REPORT action (e.g. by pressing the key that has been previously defined as the BUTTON object).

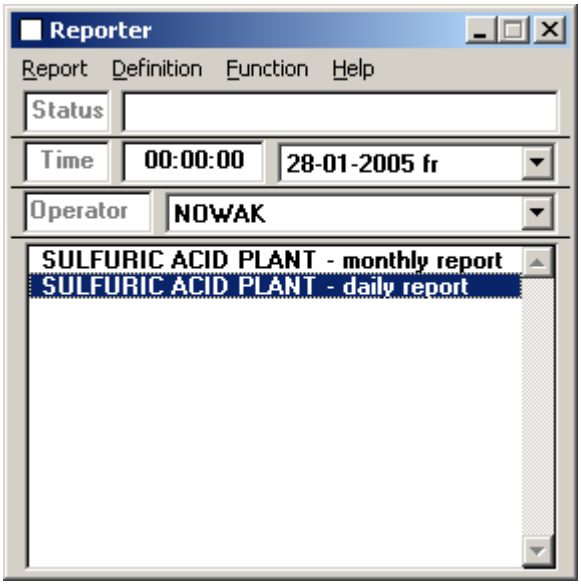


Figure 13-1. 'Reporter' Window.

In order to perform the report with use of the report window you should do the following:

- select the name of the report that is to be performed from the list of available reports,
- select the date of the beginning of the report (by default the current date is assumed),
- select the operator's name as an option (which will appear in the content of the report),
- optionally set the time of the beginning of the report (by default 00:00:00),
- optionally enter the remark (which will appear in the content of the report),
- click on the button DISPLAY in the REPORT menu.

Execution of the DISPLAY function causes checking whether the selected report has been performed before and stored on the disk. If it is the case, the report will be displayed on the screen in the specially created window.

Otherwise, the report will be calculated and stored on the disk before displaying. The reports that do not fit the screen can be scrolled within the window. The number of windows that can be simultaneously open has been limited to eight. The example of such window is shown below.

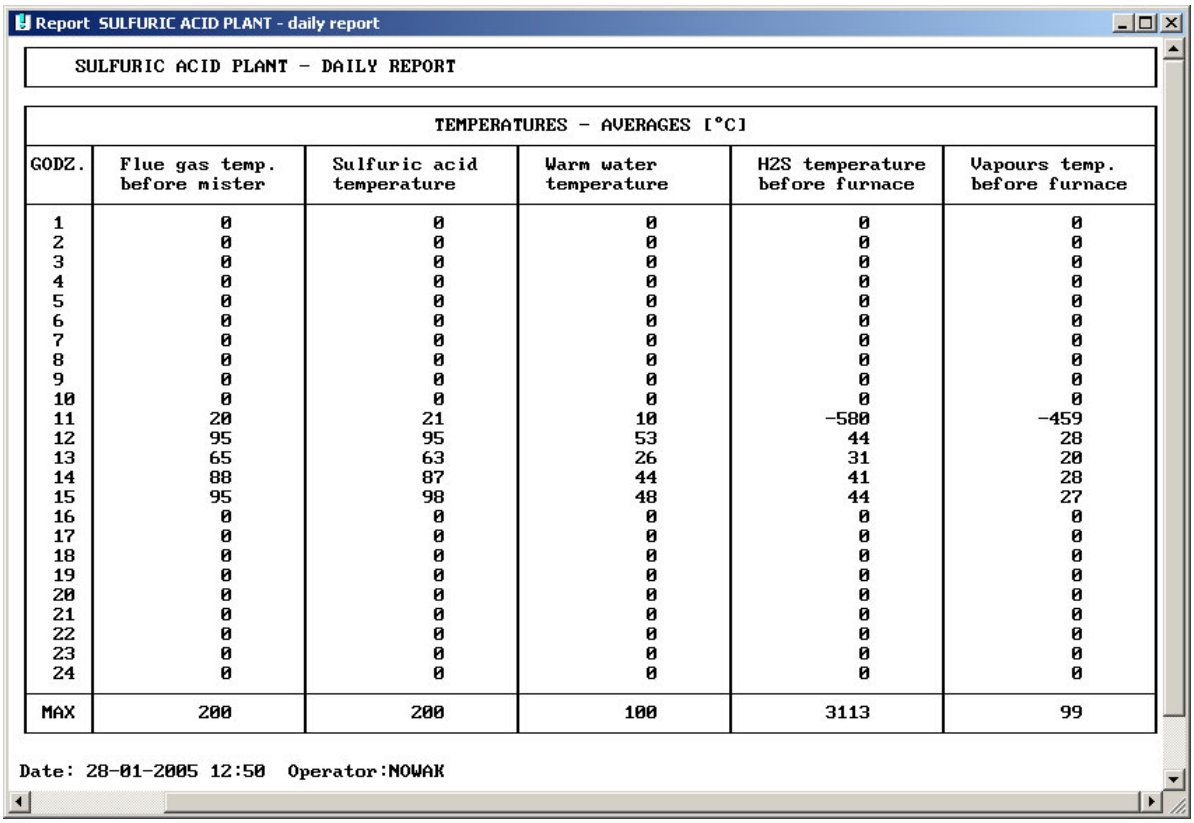


Figure 13-2. The Window with Exemplary Report.

The effect of calculating the individual report without using the report window and avoiding to display it, can be obtained by means of the action MAKE_REPORT with parameters defining:

- name of the file containing definition of the report,
- starting time of report,
- operator's name (option),
- content of the note (option).

In order to perform and display the report you should execute the SHOW_REPORT action. Moreover, this action provides possibility to delete the report from the disk after having it displayed.

For users that like to print the report without displaying it, the action PRINT_REPORT is provided.
All actions may be executed by pressing the appropriately defined BUTTON object. Moreover, they can be used in the scheduler.

Usage of the report window provides larger possibilities, than only calculating and displaying reports. Functions of the report module have been subdivided onto four groups: REPORT, DEFINITION, FUNCTION and HELP. Functions that belong to the REPORT menu enable managing of reports - calculation, deleting etc. Functions that belong to the DEFINITION menu enable managing of report definitions - checking and correcting them. The FUNCTION menu enables only to enter a note that is to be inserted into the report, whether the report is defined in appropriate manner (has got the **%report%** marker in the content of the format instruction – *see*: *13.4. ASTER - the Language for Report Definition*).

The *Status* box informs about the progress of report calculation. The report may be calculating, may be waiting in a queue for calculation or calculation may be completed. The report window cannot be closed until calculation of the report is completed.

The **REPORT** menu of the report window includes the following functions:

DISPLAY - enables displaying of the report in the window, whether the report was not stored on the disk it has to be previously calculated.

UPDATE - is similar to DISPLAY, the difference is that the reports that do not cover the full period that have been prepared for, will be recalculated. For example, if the report related to the full current day will be calculated at 10.00 p.m. and stored on the disk, execution of the DISPLAY function at 12.00 p.m. will cause displaying of the report containing data collected before 10.00 p.m. In this case the UPDATE function is to be executed due to recalculation, so the report will contain data collected until 12.00 p.m. For the reports related to the fully finished days there are no difference between the DISPLAY and UPDATE functions.

RECALC - function deletes a report from the disk (whether it was previously calculated), then the report is calculated again and displayed.

PRINT - function causes calculation of a report, then it is printed out on the printer being selected at the moment. Applying of printer control codes (ctrl command or print marker) is allowed.

DELETE - function enables removing of the report. Confirmation of this operation is required (see the dialog box below).

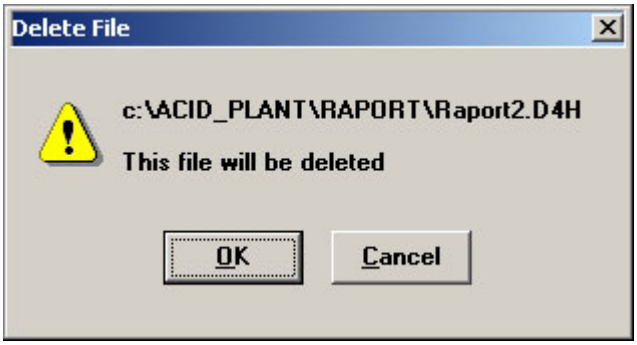


Figure 13-3. The Window with the Information on Deleted Report.

COPY - function enables copying the file that contains a report. The destination disk drive and directory has to be specified.

DIRECTORY - function enables changing of the directory, where report definitions are read. Based on the contents of them the list of report names is created in the report window. After having the directory changed, the mentioned list will be created again. Storage of reports in various directories enables subdividing the reports onto groups. The directory selection window is presented below.

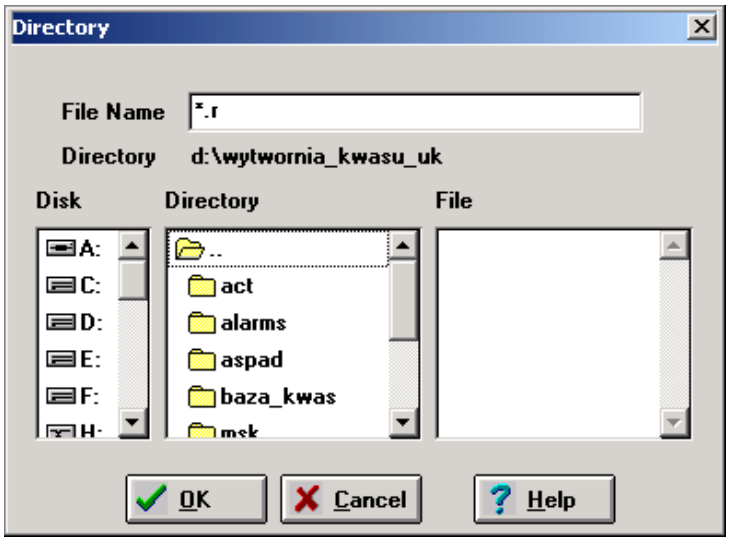


Figure 13-4. The Window for Report Directory Selection.

PRINTER - function enables selecting of the default printer.
EXIT - function enables closing of the report window.

The DISPLAY, UPDATE, RECALC, PRINT, DELETE and COPY functions need the name of the report to be selected from the report list and the report date to be set. Due to set the report date the list of dates should be previously dropped down and scrolled (if the desired date is not displayed on the list) and then the appropriate date is to be selected.

Additionally, in the same way, the operator's name, if it is going to be included in the report contents, can be selected.

The **DEFINITION** menu of the report window includes the following functions:

CHECK - function enables checking of validity of report definition. This function should be executed every time, if any modification of report definition have been carried out. The function of translation of the source report is then called. After translation completion the window containing either "no errors" information or list of detected errors with their occurrence places is displayed. More information about detected errors is included in description of the ASTER - the language of report definition.

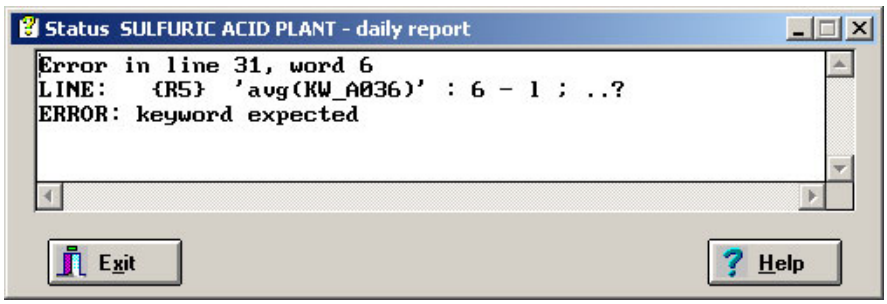


Figure 13-5. The Window of Status of Report Definition Correctness.

EDIT - function calls the text editor that enables modification of the report definition. The report that has been selected before execution of that function is automatically opened in the edition window. The edition menu provides typical function enabling creation of a new file, opening of existing file for edition, saving the corrected content of the file on a disk, deleting a file and completion of edition. The editor called by the EDIT function can also be called from the menu TOOLS of the Designer and used, for instance, for edition of initialization files (*.ini). In this case the edited file is not opened automatically, as it is executed for correction of report definition. At the time only one report can be modified. While more reports are to be displayed the VIEW function has to be used.

NOTE *The report modul uses the types in the 852 code page to ensure compatibility with DOS version in Windows 2000 / NT 4.0. In connection with that, the report editor, called by the command EDIT of the report window, set characters in the LATIN II standard. One should remember it and don't use it for changing other text file – that is theoretically possible thanks to the editor command FILE.OPEN. If one woud like to use the editor for editing Windows system text files, then the editor should be startup by command TOOLS.TEXT EDITOR…. Then polish characters will be inserted in the 1250 code page – the proper for the Windows system. Such a solution enables the possibility of reading the reports of older asix system version for DOS and the current one for the Windows.*

The edition window with report definition is shown below.

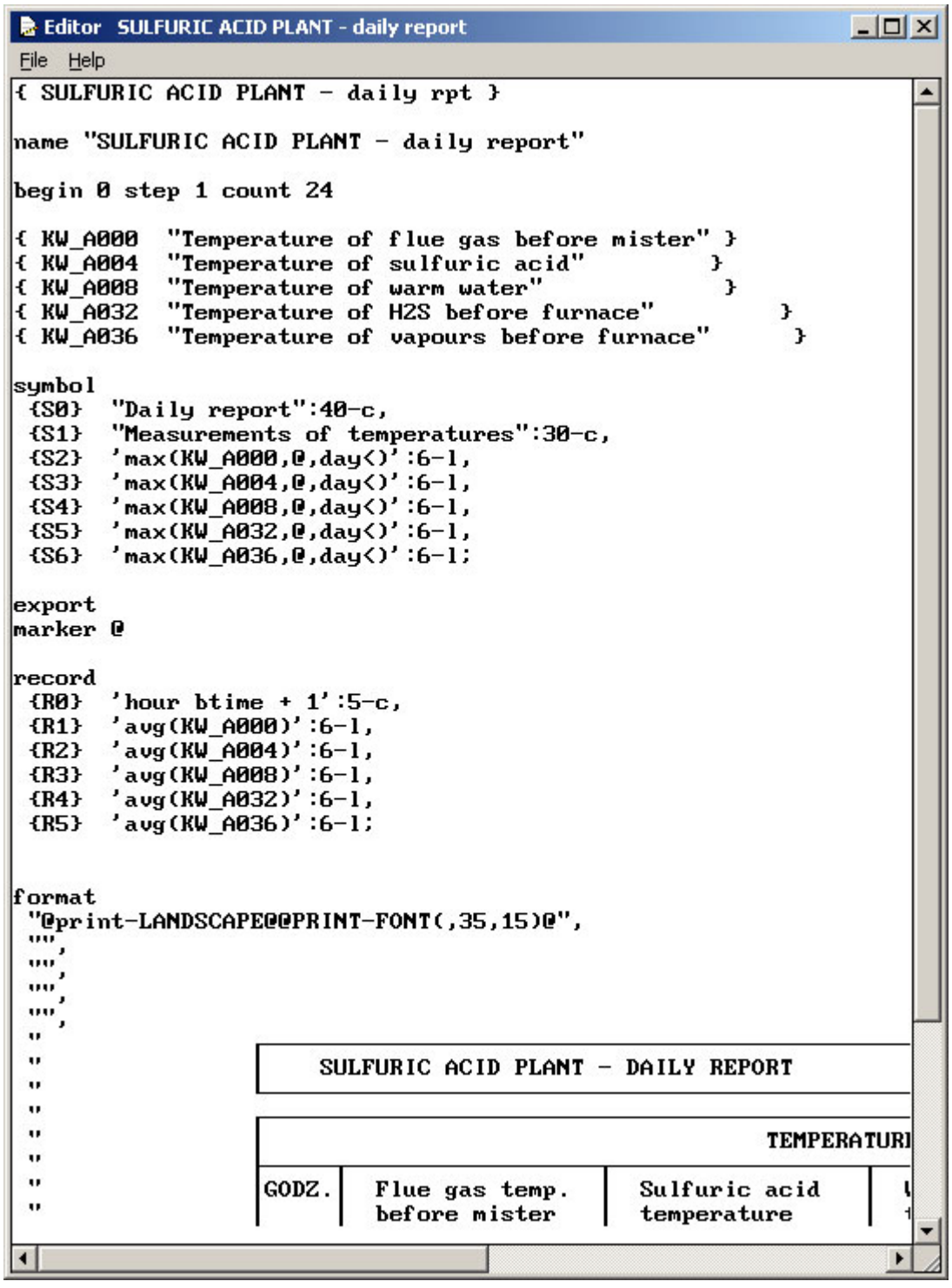


Figure 13-6. The Window of Report Definition Editor.

VIEW - function enables displaying the report definition, but no correction is possible. It is destined for comparison of report definitions.

FUNCTION menu contains two options:

NOTICE - function enables to enter the text of the note. This text is to be inserted in the report contents in place of %remark% instruction located in 'format' part of the report definition.

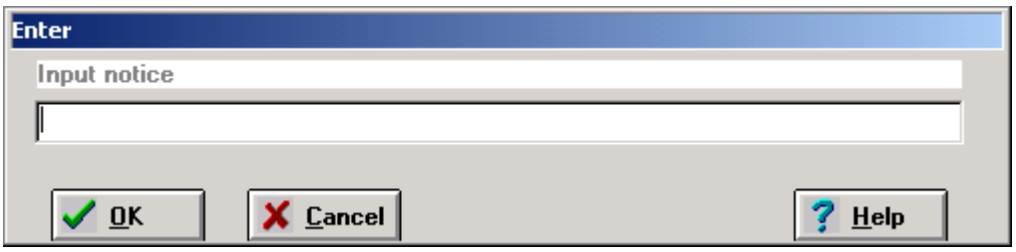


Figure 13-7. The Window for Inputting a Notice into a Report.

SET DATE - enables to select any date even not existing on the standard list of dates.

Figure 13-8. The Window for Date Setting of Report Generation.

13.1.1. Printing Reports

Apart from printing the reports by means of:

- the PRINT function of the report window, the reports can be printed out after having them displayed in the window for report displaying;
- the PRINT function is included in the menu placed under the system button of the window. this function is provided with two options:

- printing the hardcopy of a screen or a window (function Print Screen);
- printing the report as a text file (function Print Text).

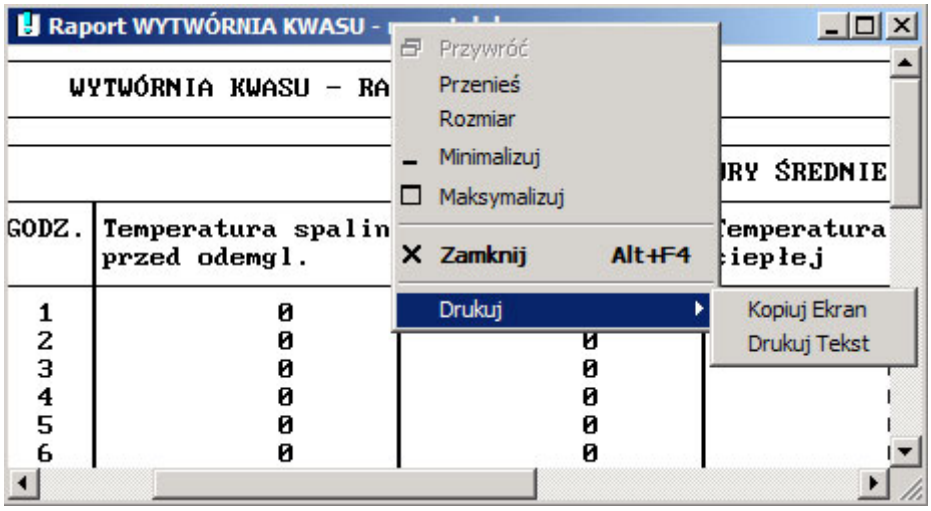


Figure 13-9. The Menu of the Report Displaying Window.

The PRINT TEXT function prints the report in the same form as the it is stored on the disk and displayed in the presentation window. It includes only report text, but does not include printer formatting.

Apart from the two functions mentioned above, printing the reports is also possible by means of the PRINT function included in the REPORT menu of the report window. The reports printed in this way are not displayed on the screen. The basic mode of report printing in the current software version is a graphics mode with a use of all mechanisms made available by the Windows system. The way of printout formatting is independent of used printer. The report content is scaled automatically to fit it to the page width. It is possibility to format the printout by means of markers PRINT-XXXX of Aster language.

Older system versions printed the reports in a text mode. Formatting was performed by means of sending appropriate codes to a printer, if they were defined in the initialization file and used in the report definition with use of **ctrl** instruction. Different types of printers may require using different control codes for the same function. For this reason, when moving an application on a computer system provided with a printer of another type, it was necessary to take care of writing, to the initialization file, proper control codes for printers corresponding with them. In case of necessity to preserve compatibility to an application made for a previous version of the **asix** system, you should select the text printout mode by declaring the ***Report in text mode*** parameter:

Architect > *Fields and Computers* > *Printout* module > *Report printing* tab

13.1.2. Handling Dialogs

By means of the DIALOGS WINDOW command in the TOOLS menu of the Control Panel menu or the Designer Window the user is able to calculate some non-typical, hardly ever used parameters and coefficients. With use of the window displayed after having the command selected, the user is allowed to:

- create and calculate expressions of the ASTEL language,
- display and modify beginning and ending moments,
- check reasons of errors that have occurred during evaluation,
- correct expressions that have been previously defined.

All information about '*Dialog Panel'* window – *see: 9.9. Using Expressions.*

13.1.3. Operator Actions

MAKE_REPORT
SHOW_REPORT
PRINT_REPORT
REPORTS

See: 16.2Description of Actions.

13.1.4. Report System Handling

Parameters of the reports system are set with use of Architect.

See: Architect user's manual, chapter. *3.13. Declaration of report definition handling*.

13.2. Designing Reports

Reports in the **asix** system are defined by means of the ASTER language. The definition written in this language is further interpreted by the program. As the result of interpretation the report is stored in the disk file. Calculation of the report is executed as result of interpretation of the set of formulas that define the report and that are written in the **record** instruction. The mentioned formulas are written in the ASTEL language, which enables definition of arithmetical expressions operating with values of process variables.

The method of report creation will be described in the further part of this chapter. However, in order to make description simply, the detailed syntax of the used statements of ASTER and ASTEL languages will be omitted. If analysis of the example reports will make troubles or will be hardly understood, more detailed information can be found in the appropriate sub-chapters going just after the main chapter and devoted to particular subjects.

The certain group of instructions must occur in each record definition. The other instructions are optional. The following instructions are included in the first group: the **name** instruction, that defines name of the report being displayed by means of the menu of the report window, the **record** instruction that defines formulas and the **format** instruction that establishes how does the report look like.

Having considered the above, definition of the report should be of the following form (a comment denotes optional instructions or groups of instructions).

name <*name*>
record <*arguments*> / or symbol <*arguments*>
{other instructions}
format <*arguments*>

The individual instructions can be ordered in any sequence. There is only one exception to this rule, i.e. the **format** instruction should follow the **record** one, as well as the instructions **header, symbol, delimiter** and **marker**, whether they occur in the definition. Due to the above, the **format** instruction usually is used as the last one.

13.2.1. The Simplest Report

So as the above information has been quoted, definition of the first simplest report can begun. The report should consist at least of three instructions: **name, record** and **format**. The first one is used for assignment of the own name to the report. The name is used in further for identification of the report. Next, the **record** instruction specifies the list of formulas used for calculations together with information about format of output data (field width, number of fraction digits, style etc.). For the simplest case, this list consists of only one component and the expression (formula) can be very simply (e.g. the constant). The **format** instruction defines form of the report (headers, frames, etc.).

EXAMPLE

As referred to the above, definition of the simplest report can be as follows:

```
name "simple"              {the name "simple" included into the menu}
record '1':1;              {calculation of the formula '1'}
format "%r0%";             {value of 1 included into the report}
```

As a result of execution of the above definition, the formula '1' is to be calculated. Its result (value of 1) will be in further included into the report file. The obtained report consists of only one digit and is independent on status of process variables.

13.2.2. Files

Both definitions of reports and the reports themselves are stored in disk files. Files that include definitions should have the **.r* extension. The file that includes a report has the same name as the report definition file, but it differs only with extension. Every definition may be referred to many report files, because the definition determines the report exact to the day of beginning of it. Therefore, the report files have various extensions, which enable to determine the day that the report refers to. For instance, if definition presented above were stored into the *simply.r* file, the related report would be located in the file *simply.nnn*, where *nnn* denotes the date of the report. The date is to be established by making of the report (even if a content of the report is independent of the date). The files that include reports and their definitions are stored in the same directory).

Date of the report can be determined based on the file extension. The digits of the extension in the order of appearance denote respectively: year-1990, month and day. For notation of numbers greater than 10 the consecutive letters starting from "a" are used.

EXAMPLES

report.111	*-1*	*-*	*(1) of January (1) 1991 (1) year*
report.54b	*-11*	*-*	*(b) of April (4) 1995 (5) year*
report.bbb	*-11*	*-*	*(b) of November(b) 2001 (b) year*

13.2.3. Description of Fields

In all the reports, after formulas used for **record** and **symbol** instructions or strings applied to **header, input** and **symbol** instructions, descriptions of fields can appear. They will be hereby discussed. Every description of the field can consist of sequence of maximum two numbers preceded by colons, then one of the following letters: **L, R, C** or **Z** preceded by the '-' (minus) sign (so called style) can appear as an option. The letters denote:

- the style defines explicit location of characters in the field (L- filling with spaces on the left, R - on the right, C - on the both sides, Z - filling with zeroes on the left);
- the first number defines width of the field designed either for a number or for a string; whether length of the string is less than the defined field width, spaces are added to fill the whole field; spaces are added depending on the selected style on the left (L), on the right (R) or on both sides (C); adding of non-significant zeroes on the left is possible for numbers (Z); whether length of a string exceeds maximum field width, the string is truncated; whether the number does not fit within the specified field, it is replaced by the string "?FRM", that denotes format error; the error message can also be truncated to fit within the selected field width;
- whether field width is missed, the string comes out in its own form (without filling or truncating); whether field width is missed for numbers, output of numbers is impossible (but formulas are evaluated); it can be used in practice in case of assignment of value of the formula

to an auxiliary variable; output of such a variable is not necessary, but it can be used in further for calculation of other formulas;

- the second number denotes number of fractional decimal digits that is to be available in the output variable; whether the second number is omitted, output of the integer part is carried out; if the sum of number of integer decimal digits plus one (position of the decimal point) and fractional decimal digits exceeds the field with, the error occurs; in case of strings, the second number denotes how many characters should be sent to the output; if omitted, the full string will be outputted (whether it fits within the given field.

It is possible to modify standard formats denoted with letters **L**, **R**, **C** and **Z** by adding additional declarations included in a group of characters controlling printout of warnings. After a character determining a style, i.e. **L**, **R**, **C**, **Z,** one should place characters controlling the printout in case of occurrence of an error or a warning of a specified type. Following characters are allowed:

- **F** – reference to the future;
- **H** – a hole occurred;
- **B** – time back occurred;
- **N** – no data (but the variable is declared);
- **R** – no answer from archive occurred.

Position of listed characters causes that the character „?”, informing about an error, will not be displayed together with the value. Modifiers of characters determining a style may be combined. To increase the legibility it is recommended writing the modifiers with lower case letter to separate them visually from a style marker. It is illustrated by the below example.

EXAMPLE

If the format is declared as below,

:8:3-Lfh,

and if a hole in archive data or a reference to the future occurs, the character **?** will not be output together with a number, and if a string **0?** should be output additionally then an empty place will be output.

13.2.4. Definition of Reports

Report Formatting
The Report with a Border
The Report with a Border and a Header
Period of the Report
One-Variable Report
Usage of Variables
Report with Variables
Global Formulas
The report with Global Formulas
Alarm Reports
The "Post Mortem" Report
Reports with Inserted Fields
Exported Reports
Exported Reports in "Delimited with" Format
Reports with Unknown Number of Lines
Report about Limit Exceeding

Based on a one of previous chapters one should know how to define the simplest report. The report presented before included only one number written to the file. But usually reports are much more sophistic. Most often they have a form of a table that consists of certain number of rows and defined number of

columns. The individual lines are referred to consecutive periods and the columns relate to the selected technological coefficients. The table is usually titled, has a legend, operator's name, etc. The ASTER language enables creation of such reports. Features of the language will be hereby consequently discussed.

Report Formatting

Thanks to implementation of the **format** instruction creation of borders, descriptions, legends, etc. is possible. Output of consecutive lines of the report is controlled by content of consecutive lines, where the **format** instruction has been used. Every line of the **format** instruction can include ordinary characters or markers. Characters are inserted into the report. After having met a marker the appropriate action is carried out. Such an action can be, for instance, evaluation of a formula defined by the **record** instruction, and inserting of the result into the report. Due to make it possible, the marker **%rN%** is assigned to every next formula from the **record** instruction (where *N* stands for number of the formula - starting from zero). In the same way, strings declared by the **header** instruction can be used. Referring to them is possible by means of the **%hN%** markers. Only one line is allowed to include the **%rN%** markers in the **format** instruction, but it can be interpreted many times (**count** times). The examples illustrate, how to add a border to the report taking advantage (or not) of the **header** instruction.

EXAMPLE

The Report with a Border

Three process variables A, B and C have been defined. Average values of them for three consecutive one hour long periods (from the established moment) are to be outputted in the report. The field of 8 digits, with 3 fractional ones, is assigned to every number. Additionally, the variables A, B and C, that stand for pressure, temperature and volume, should be described and the time referred to every line is to be marked as well. The report should be composed with the border around.

```
name "border"                    / name
begin 12                         / reporting is
begun at 12 o'clock
step 1                           / period of 1 hour (by default)
count 3                          / three lines
record
{%r0%} 'hour btime':8-c,         / marker %r0% starting time (hour)
{%r1%} 'avg a':8:3-c,            / marker %r1% calculation of /average
                                   value for the A variable
{%r2%} 'avg b':8:3-c,            / marker %r2% calculation of /average
                                   value for the B variable
{%r3%} 'avg c':8:3-c;            / marker %r3% calculation of /average
                                   value for the C variable
format
"+--------+--------+--------+-------+",
"¦  hour  ¦ press. ¦  temp. ¦  vol. ¦",
"+--------+--------+--------+-------¦",
"¦%r0%¦%r1%¦%r2%¦%r3%¦",     /interpreted count times
"+--------+--------+--------+-------+";
```

The **header** instruction hasn't been used in the example. After the report based on the above definition has been calculated, the following report would be created:

```
+--------+--------+--------+--------+
¦  hour  ¦ press. ¦  temp. ¦  vol.  ¦
+--------+--------+--------+--------+
¦   12   ¦ 15.300 ¦ 88.000 ¦ 33.333 ¦
¦   13   ¦ 16.850 ¦ 91.500 ¦ 29.200 ¦
¦   14   ¦ 13,333 ¦ 92.245 ¦ 21.880 ¦
+--------+--------+--------+--------+
```

EXAMPLE

The Report with a Border and a Header

Three process variables A, B and C have been defined, average values of theirs for three consecutive periods one hour long (from the established moment) are to be outputted in the report. The field of 8 digits, with 3 fractional ones, is assigned to every number. Additionally, the variables A, B and C, that stand for pressure, temperature and volume, should be described and the time referred to every line is to be marked as well. The report should be composed with the border around and the header definition is to be used. The appropriate report definition can be as follows:

```
{Definition of the report with the border and the header}
name "border2"              / name
begin 12                    / reporting is begun at 12 o'clock
step 1                      / period of 1 hour (by default)
count 3                     / three lines
header
{%h0%} " hour":8-c,         / marker %h0% time description
{%h1%} " press."8-c,        / marker %h1% the A variable /description
{%h2%} " temp."8-c,         / marker %h2% the B variable /description
{%h3%} " vol."8-c;          / marker %h3% the C variable description
record
{%r0%} 'hour btime':8-c,    / marker %r0% starting time (hour
{%r1%} 'avg a':8:3-c,       / marker %r1% calculation of average /value
                              for the A variable
{%r2%} /avg b':8:3-c,       / marker %r2% calculation of average /value
                              for the B variable
{%r3%} 'avg c':8:3-c;       / marker %r3% calculation of /average value
                              for the C variable
format
   "+----------------------------------+",
   "¦%h0%¦%h1%¦%h2%¦%h3%¦",
   "+--------+--------+--------+--------¦",
   "¦%r0%¦%r1%¦%r2%¦%r3%¦",   / interpreted count times
   "+----------------------------------+";
```

The **header** instruction has been used in the above report. After the report based on the above definition is calculated, the report identical to the one of the previous example will be created.

Period of the Report

Period, that the report is referred to, is declared by means of **begin, step** and **count** instructions. The **step** instruction defines time interval referred to one line of the report. The argument of the instruction can define time intervals varying from seconds to many days (note: calculation time is then prolonged). Period of the report is evaluated as product of number of lines (defined by the **count** instruction) and the time quantum assigned to one line.

Initial moment for the first line is established by adding the time defined by the **begin** instruction to the starting time (having been set within the report dialog window). Terminating moment for the first line is calculated by summing up the initial moment and the time interval defined by the **step** instruction. For every next line the terminating moment of the previous one becomes the initial moment for the current one and the new terminating moment is calculated in the same manner as for the first line. The moments established in this way are called **the default moments** and are used for definition of periods used for evaluation.

The situation described above occur, if the initial and terminating moments are never directly established in formulas for used evaluators. It is not necessary - the user is allowed to insert any absolute and relative moments achieving in this way ability to create reports, where different periods, referred or not to the default time, can occur in the same line. It will be explained by means of the following examples.

EXAMPLE

One-Variable Report

Only one process variable A (temperature) has been defined, average values of it are to be outputted in the report. The average values should cover the 15-minutes (quarter) time intervals. The average values for four consecutive quarters are to be outputted by each line. Continuous lines should refer to the three adjacent one-hour periods (starting from the established moment of time). The field of 8 digits, with 3 fractional ones, is assigned to every number. The appropriate report definition can be as follows

```
{Definition of the one-variable report}
name "1-variable"                          / name
begin 12                                   / reporting is begun at 12 o'clock
step 1                                     / period of 1 hour (by default)
count 3                                    / three lines
record
{%r0%} 'hour btime':8-c,                   / starting time (hour)
{%r1%} 'avg(a,@,<0\45)':8:3-c,             / calculation for the first /quarter
{%r2%} 'avg(a,>0\15,<0\30)':8:3-c,         / calculation for the second /quarter
{%r3%} 'avg(a,>0\30,<0\15)':8:3-c,         / calculation for the third /quarter
{%r4%} 'avg(a,>0\45)':8:3-c;               / calculation for the fourth /quarter
format
   "+--------+--------+--------+--------+--------+",
   "¦  hour  ¦1-quart.¦2-quart.¦3-quart.¦4-quart.¦",
   "+--------+--------+--------+--------+--------+",
   "¦%r0%¦%r1%¦%r2%¦%r3%¦%r4%¦", / interpreted count times
   "+--------+--------+--------+--------+--------+";
```

In the example the default, one-hour long period has been divided into 4 quarters by means of applying of relative moments that shift both the initial moment and the terminating one for every evaluator of the average. After the report based on the above definition has been calculated, the following report would be created:

hour	1-quart.	2-quart.	3-quart.	4-quart.
12	15.300	15.900	15.500	15.200
13	15.100	15.000	15.200	15.000
14	15.300	15.500	15.800	15.400

Usage of periods of the types "from the beginning of the day to the default moment", "from the default moment to the defined moment" and "from the defined moment during the established time interval", etc., is also possible.

Usage of Variables

The valuable abilities for report creation, where some columns depend on the other ones can be incorporated by applying the variables. By means of them some values, that are necessary for calculation within other columns, can be kept. The above will be illustrated by the following example.

EXAMPLE

Report with Variables

Two process variables P and Q have been defined (standing for active and reactive power respectively). Average values of them for three consecutive one hour long periods (from the established moment) are to be outputted in the report. Moreover, the average value of the apparent power S is to be also calculated in the next column. The field of 8 digits, with 3 fractional ones, is assigned to every number. The appropriate report definition can be as follows:

```
{Definition of the report with variables}
name "with variables"                  / name
begin 12                               / reporting is begun at 12 o'clock
step 1                                 / period of 1 hour (by default)
count 3                                / three lines
record
{%r0%} 'hour btime':8-c,               / marker %r0% starting time
                                         (hour)
{%r1%} 'p=avg p':8:3-c,                / marker %r1% calculation of
                                         average /value for the P variable
{%r2%} 'q=avg q':8:3-c,                / marker %r2% calculation of
                                         average /value for the Q variable
{%r3%} 's=sqrt(p^2+q^2)':8:3-c;        / marker %r3% calculation of the s
                                         /variable
format
   "+-----------------------------------+",
   "¦ hour ¦ P[W] ¦ Q[VAr] ¦ S[VA] ¦",
   "+--------+--------+--------+--------+",
   "¦%r0%¦%r1%¦%r2%¦%r3%¦", /interpreted count times
   "+--------+--------+--------+--------+",;
```

After the report based on the above definition has been calculated, the following report would be created:

```
+-----------------------------------+
¦ hour ¦ P[W] ¦ Q[VAr] ¦ S[VA] ¦
+--------+--------+--------+--------+
¦  12  ¦ 10.000 ¦ 2.000 ¦ 10.198 ¦
¦  13  ¦ 12.000 ¦ 2.500 ¦ 12.258 ¦
¦  14  ¦ 13.000 ¦ 2.500 ¦ 13.238 ¦
+--------+--------+--------+--------+
```

Calculation of the appeared power for the above example could be executed without usage of variables. It would be enough to put in:

sqrt((avg p)^2+(avg q)^2)

Although, the above method would cause double calculation of average values of the *P* and *Q* variables, that leads to more than twice increasing of report calculation time.

For the example presented above the order of evaluation is the same as the order of appearance of formulas in the **record** instruction, i.e. the value of P is calculated first, then Q as the second and S is calculated last for the period from 12 o'clock to 1 p.m. (and later for consecutive periods).

Global Formulas

There is a possibility to introduce the report lines containing results of formulas that do not appear in the **record** instruction. Such lines are not interpreted **count** times. There is the **symbol** instruction that enables declaration of global formulas and strings. The default initial and terminating moments for formulas declared by means of the **global** instruction are equivalent to the moments of starting and completion of the report. Usage of the **symbol** instruction for a string declaration provides possibility of its style and field width declaration.

Markers of **%sN%** type are assigned to the elements being declared by means of the **symbol** instruction. The following example is explaining how to use global formulas.

EXAMPLE

The Report with Global Formulas

Three process variables A, B and C have been defined, average values of them for three consecutive one hour long periods (from the established moment) are to be outputted in the report. The field of 8 digits, with 3 fractional ones, is assigned to every number. Additionally, the variables A, B and C that stand for pressure, temperature and volume, should be described, and the time referred to every line is to be marked as well. The value of variance of the A, B and C variables for the whole three-hours period should be outputted in the last line of the report. Definition of the report can look as follows:

```
{Definition of the report with global formulas}
name "border "                  / name
begin 12                        / reporting is begun at 12 o'clock
step 1                          / period of 1 hour (by default)
count 3                         / three lines
symbol
{%s0%} "variance.":8-c,         / marker %s0% description
{%s1%} 'var a':8:3-c,           / marker %s1% calculation of variance of the
                                /A variable
{%s2%} 'var b':8:3-c,           / marker %s1% calculation of variance of the
                                /B variable
{%s3%} 'var c':8:3-c;           / marker %s1% calculation of variance of the
                                /C variable
record
{%r0%} 'hour btime':8-c,        / marker %r0% starting time (hour)
{%r1%} 'avg a':8:3-c,           / marker %r1% calculation of average value
                                /for the A variable
{%r2%} 'avg b':8:3-c,           / marker %r1% calculation of average value
                                /for the B variable
{%r3%} 'avg c':8:3-c;           / marker %r1% calculation of average value
                                /for the C variable
format
   "+----------------------------------+",
   "¦  hour  ¦ press. ¦  temp. ¦  vol.  ¦",
   "+--------+--------+--------+--------+",
   "¦%r0%¦%r1%¦%r2%¦%r3%¦",/interpreted count times
   "+--------+--------+--------+--------+",
   "¦%s0%¦%s1%¦%s2%¦%s3%¦",/interpreted only once
   "+--------+--------+--------+--------+";
```

After the report based on the above definition has been calculated, the following report would be created:

```
+---------------------------------+
¦  hour  ¦ press. ¦  temp. ¦  vol.  ¦
+--------+--------+--------+--------+
¦   12   ¦ 15.300 ¦ 88.000 ¦ 33.333 ¦
¦   13   ¦ 16.850 ¦ 91.500 ¦ 29.200 ¦
¦   14   ¦ 13.333 ¦ 92.245 ¦ 21.880 ¦
+--------+--------+--------+--------+
¦variance¦ 0.333  ¦ 2.240  ¦ 8.800  ¦
+--------+--------+--------+--------+
```

Alarm Reports

Some types of reports are calculated periodically. Other reports may be calculated at emergency situations (breakdowns, setting in motion). The requirement towards the report is to cover the period just before the event took place. Reports of such type are often called as "post mortem" ones. They are implemented as a special type of reports, because they are available only one time after the breakdown has occurred. In the **asix** system such reports can be calculated many times, they can even be defined after the breakdown took place. The report period can be set separately for every formula calculated within the report. The above is illustrated by the following example.

EXAMPLE

The "Post Mortem" Report

Three process variables A, B and C have been defined, in the report maximum values of them are to be outputted for periods that precede the breakdown (calculation of the report). The periods for A, B and C variables are respectively 30, 60 and 90 minutes. The field of 8 digits, with 3 fractional ones, is assigned to every number. Additionally, the variables A, B and C, which stand for pressure, temperature and volume, should be described. Definition of the report can look as follows:

```
{Definition of the alarm report}
name "post mortem"                  / name
record
{%r0%} 'max(a,&,&0\30)':8:3-c,      / marker %r0% calculation of
                                      maximum /value for the A variable
{%r1%} 'max(b,&,&1)':8:3-c,         / marker %r1% calculation of
                                      maximum /value for the B variable
{%r2%} 'max(c,&,&1\30)':8:3-c;      / marker %r2% calculation of
                                      maximum /value for the A variable
format
   "+--------+-----------------+",
   "¦ bkdown ¦ %date% %time% ¦",
   "+--------+--------+--------+",
   "¦ press. ¦  temp. ¦ volume ¦",
   "+--------+--------+--------+",
   "¦%r0%¦%r1%¦%r2%¦",
   "+--------+--------+--------+";
```

After the report based on the above definition has been calculated, whether the breakdown took place on the 15th of July 1993 at 3:20 p.m., the following report would be created:

```
+-------------------------+
¦ bkdown ¦ 15-07-93 15:20  ¦
+--------+--------+--------+
¦ press. ¦  temp. ¦ volume ¦
+--------+--------+--------+
¦ 25.300 ¦ 98.000 ¦ 23.700 ¦
+--------+--------+--------+
```

The markers **%date%** and **%time%** have been used in the report.

For typical situations the alarm report can be calculated manually by an operator, after having pressed the appropriate key (the BUTTON object), or automatically, by use of the events scheduler.

Reports with Inserted Fields

Frequently there is the need to insert into the content of the report some additional information that isn't stored in the system (number of order, symbol of product, customer identification). An operator can insert those data into the report while creating it. The **input** instruction is used for this purpose. Markers of **%iN%** type are assigned to the elements being declared by means of the **input** instruction. Every component declared by the **input** instruction includes both a question (prompt) that is outputted during interpretation of the marker (there can be an empty question), and specification of the area dedicated for the answer. The following example is explaining how to use inserted fields.

EXAMPLE

The Report with Inserted Fields

Two process variables A and B have been defined, the average values of them for three consecutive one hour long periods (from the established moment) are to be outputted in the report. The field of 8 digits, with 3 fractional ones, is assigned to every number. Additionally, the variables A and B, which stand for current and temperature, should be described as well as the time referred to every line. Moreover, every line should include name of the raw material inserted by an operator and the whole report should include number of production series. Definition of the report can look as follows:

```
{Definition of the report with inserted fields}
name "inserted"                                  / name
begin 12                                         / reporting is begun at 12
                                                   o'clock
step 1                                           / period of 1 hour (by default)
count 3                                          / three lines
input
{%i0%}  "enter the series number?" :8-c,         / marker %i0% entering the
                                                   /series number
{%i1%}  "enter name of material ?" :8-c;         / marker %i1% entering
                                                   /name of material
record
{%r0%} 'hour btime':8-c,                         / marker %r0% starting time
                                                   (hour)
{%r1%} 'avg a':8:3-c,                            / marker %r1% calculation of
                                                   average value /for the A
                                                   variable
{%r2%} 'avg b':8:3-c;                            / marker %r2% calculation of
                                                   average value /for the B
                                                   variable
format
"+-----------------------------------+",
"¦  Series number       %i0%        ¦",
"+-----------------------------------+",
"+--------+--------+--------+--------+",
"¦  hour  ¦material¦ current¦  temp. ¦",
"+--------+--------+--------+--------+",
"¦%r0%¦%i1%¦%r2%¦%r3%¦",
"+--------+--------+--------+--------+",
```

After the report based on the above definition has been calculated, whether the operator's answers will be, for instance: "Z2511", "steel", "steel" "zinc" the following report would be created:

```
+-----------------------------------+
¦  Series number       Z2511        ¦
+-----------------------------------+
+--------+--------+--------+--------+
¦  hour  ¦material¦ current¦  temp. ¦
+--------+--------+--------+--------+
¦   12   ¦ steel  ¦ 38.000 ¦ 33.333 ¦
¦   13   ¦ steel  ¦ 41.500 ¦ 29.200 ¦
¦   14   ¦  zinc  ¦ 12.245 ¦ 91.880 ¦
+--------+--------+--------+--------+
```

Exported Reports

Frequently there exists the need to make data located in the report available to other programs (databases, spreadsheets). Data are exported in text form and can be read by databases or spreadsheets either as numbers or as texts. In case of exporting the reports with conversion (texts to numbers) the **export** instruction is recommended. This instruction allows putting in fields of the reports only such numbers, which format is accepted during conversion. Therefore, the error messages (of ?ARC type) will not be visible (the related

fields will be filled with spaces). Adding the question character while non-critical errors occur will also be neglected (sometimes it causes impossibility of conversion). Export of reports in text form is also possible. Although, further processing of such data is very complicated (the error messages should be interpreted and numbers with question character would have to be separately converted).

In case of export of data to relational databases (dBase, FoxBase, Clipper, CodeBase) the problem is in principle restricted to matrixes of numbers. Headers and description can also be exported to spreadsheets (Quattro) (however, borders should be drawn by means of built-in mechanism).

The example of exporting of the report to the database of dBase, FoxBase and Clipper is going to be presented below. For this case, the report should be prepared in SDF format (text records of fixed length ended by CR-LF characters) or as records with individual fields separated by declared separators (DELIMITED WITH).

For the second case description of report fields needn't match description of record fields within the database (the only requirement is that record fields of a database would be long enough to fit the appropriate values of a report). The above is illustrated by the following example.

EXAMPLE

Report that is to be Exported in the "Delimited with" Format

Three process variables A, B and C have been defined, average values of them for three consecutive one hour long periods (from the established moment) are to be outputted in the report. The field of 6 digits, with 3 fractional ones, is assigned to every number. Additionally, the moment referred to every line is to be marked as well. The report is to be exported to the database (dBase). Commas should separate individual fields.

```
{Definition of the exported report in the "delimited with" format}
name "export delimited"        / name
begin 12                       / reporting is begun at 12 o'clock
step 1                         / period of 1 hour (by default)
count 3                        / three lines
export                         / conversion to numbers
record
{%r0%} 'hour btime':3-c,       / starting time (hour)
{%r1%} 'avg a':7:3-c,          / calculation of average value for /the A
                                 variable
{%r2%} 'avg b':7:3-c,          / calculation of average value for /the B
                                 variable
{%r3%} 'avg c':7:3-c;          / calculation of average value for the C
                                 variable
format
"%R0%,%R1%,%R2%,%R3%";/ delimiter - comma
```

After the report based on the above definition has been calculated, the following report would be created:

```
12, 15.300, 88.000, 33.333
13, 16.850, 91.500, 29.200
14, 13.333, 92.245, 21.880
```

The report that has been created in the above manner can be inserted into the Dbase type database by means of the instruction:

APPEND FROM <report_name> DELIMITED WITH ,

Reports with Unknown Number of Lines

The reports, that have already been presented, had one common feature - the constant, previously declared number of lines. The constant time interval was also assigned to every line. But it is not a must. Sometimes the number of lines and the time interval assigned to every line may depend on behavior of the archive variables being used in the report. This is applicable to reports that use formulas operating with time limits established by moments of exceeding of technological limits. The **under** and **over** evaluators are used for fixing the moments of limit exceeding. They are usually used in formulas that are calculated (for fixing the appropriate moments), but values of them are not shown. Additionally, the line with **%rN%** markers should include (usually at its end) the **%next%** marker that forces establishing of the new initial moment. The calculated value of the terminating moment becomes the initial one and the default terminating moment is restored again. It is illustrated by the example.

EXAMPLE

Report about Limit Exceeding

The process variable A (temperature) has been defined. Intervals, while it exceeded the technological limit of 155 °C are to be discovered. Searching should be carried out during the given one-hour long period that begins at 7 o'clock. The starting and finishing moments, duration time and maximum value are to be presented for every time interval that has been found in that manner. Additionally, an extra column should be added that include the maximum value that is shown in case, whether the whole interval falls into the given period, otherwise it contains zero. The report should be composed with the border around. The field of 8 digits, with 3 fractional ones, is assigned to every number. Absolute time values and periods are to be shown as "hour, minute, second". The appropriate report definition can be as follows:

```
{Definition of the report about limit exceeding}
name "exceeding"                / name
begin 7                         / reporting is begun at 7 o'clock
step 1                          / period of 1 hour
repeat                          / unknown number of lines
record
{%r0%} 'p=155'8:2-c,            / marker %r0% - setting the limit
{%r1%} 'o=over>a,p',            / marker %r2% - initial moment
{%r2%} 'u=under<a,p',           / marker %r2% - terminating moment
{The two above formulas do not output results - they are evaluate only}
{%r3%} 'mx=max a':10:3-c,       / marker %r3% - searching for maximum
{%r4%} 'o*u*mx':10:3-c;         / marker %r4% - maximum of x interval
format
"+--------+----------+----------+----------+----------+----------+",
"¦ limit ¦  begin   ¦   end    ¦ maximum  ¦  diff.  ¦interv-max¦",
"+--------+----------+----------+----------+----------+----------+",
"¦%R0%¦%R1%%R2% %btime% ¦ %etime ¦%R3%¦ %dtime%
```

```
¦%R4%¦%NEXT%",
"+--------+----------+----------+----------+----------+----------+",
```

After the report based on the above definition has been calculated, the report presented in the further part would be created. The attention should be paid to the form of the last line. It does not contain data about exceeding of the limit (maximum is lower than limit and exceeding time is zero).

The formulas **%r1%** and **%r0%** are evaluated without output of the results. The order of evaluation is determined by the appropriate line of the **format** instruction. The markers must occur in the given order before the **%btime%** marker.

```
+--------+----------+----------+----------+----------+----------+
¦  limit ¦   begin  ¦    end   ¦ maximum  ¦   diff.  ¦interv-max¦
+--------+----------+----------+----------+----------+----------+
¦ 155.00 ¦ 07:16:00 ¦ 07:18:00 ¦  160,130 ¦ 00:02:00 ¦  160.130 ¦
¦ 155.00 ¦ 07:22:00 ¦ 07:24:00 ¦  155,004 ¦ 00:02:00 ¦  155.004 ¦
¦ 155.00 ¦ 07:28:00 ¦ 07:34:00 ¦  170,626 ¦ 00:06:00 ¦  170.626 ¦
¦ 155.00 ¦ 07:42:00 ¦ 07:46:00 ¦  162,571 ¦ 00:04:00 ¦  162.571 ¦
¦ 155.00 ¦ 07:52:00 ¦ 07:56:00 ¦  162,815 ¦ 00:04:00 ¦  162.815 ¦
¦ 155.00 ¦ 07:56:00 ¦ 07:56:00 ¦  146,704 ¦ 00:00:00 ¦   0.000  ¦
+--------+----------+----------+----------+----------+----------+
```

Both for the case, when there is no limit exceeding and when all the values are above the limit the reports about limit exceeding produce one line of the report. The examined situation can be estimated basing on that line. For the report presented above there would be the lines with zero duration, while the initial moment is equal to the terminating one. For the first case (no exceeding) the both times are set at the beginning of the period. For the second one (the whole period is exceeded) - at the end.

13.2.5. The Examplary Report

The typical (practically applied) report will be presented below. The report is related to the metallurgical pusher furnace for heating of slabs before hot rolling. Such a furnace is subdivided into three parts fired with gas. The average gas consumption, average temperatures and air volume to gas volume ratio are the most important parameters for each zone. Moreover, the report should include the average temperature of slabs and number of heated slabs. The report should cover the full day (24 hours). Because of special time schedule in the steel works, every day (subdivided into three shifts) begins at 10 p.m. (the shifts start relatively at 10 p.m., 6 a.m. and 2 p.m.). The period of one day should be divided into 24 one-hour long intervals. Maximum gas consumption for the furnace is about 6000 [m^3/h]. The air volume to gas volume ratio varies from 5 to 8 (it should be shown with precision of two decimal fractional digits). Temperatures in individual zones of the furnace vary from 1200°C to 1300°C. Temperatures of slabs that come out from the furnace are usually few dozen lower that temperature of furnace zones. While the furnace operates at maximum efficiency, the number of slabs that come out from it does not exceed 200 pieces per hour (usually it is lower by half). The report should include the date (the starting day of the report) and operator's name.

```
{Definition of the report about limit exceeding}
 name "daily"                    / name of the report
 begin 22 step 1 count 24
 The full-day (24 hours) report, reporting is begun at 10 p.m.; one line
                                 refers one hour}
 Record
 {%R0%} 'hour btime':2,          / interval starting hour
 {%R1%} 'hour etime':2,          / interval finishing hour
 {%R2%} 'avg M6712':4,           / average gas consumption in the 1 zone
 {%R3%} 'avg M6714':4,           / average gas consumption in the 2 zone
 {%R4%} 'avg M6716':4,           / average gas consumption in the 3 zone
 {%R5%} 'avg PowGas1':4:2,       / average air/gas ratio in the 1 zone
 {%R6%} 'avg PowGas2':4:2,       / average air/gas ratio in the 2 zone
 {%R7%} 'avg PowGas3':4:2,       / average air/gas ratio in the 3 zone
 {%R8%} 'avg M7514':4,           / average temperature in the 1 zone
 {%R9%} 'avg M7515':4,           / average temperature in the 2 zone
 {%R10%} 'avg M7516':4,          / average temperature in the 3 zone
 {%R11%} 'avg M7512,v':4,        / average temperature of outgoing slabs
 {%R12%} 'diff (cntLSW,v)':3;    / number of outgoing slabs
 format
 "THE ŁABĘDY STEELWORKS                                  Pusher furnace"
 "Operator: %operator%              Date: %bdate%",
 "+-----------------------+---------------+-----------------+----------+",
 "¦from-to¦g1[m3]g2[m3]g3[m3]¦ p/g1 p/g2 p/g3
 ¦t1[°C]t2[°C]t3[°C]¦tk°C]  lk¦",
 "+-----------------------+---------------+-----------------+----------+",
 /"¦ xx-xx ¦ xxxx  xxxx  xxxx ¦ x.xx x.xx x.xx ¦ xxxx  xxxx  xxxx ¦ xxxx
 xxxx¦
 "¦ %R0%-%R1% ¦ %R2%  %R3%  %R4% ¦ %R5%  %R6%  %R7% ¦
 %R8%  %R9%  %R10% ¦ %R11%  %R12% ¦",
 "+-----------------------+---------------+-----------------+----------+",
```

The report will be composed with a border around. To average gas consumption within every zone the field of the width of four characters has been assigned (maximum gas consumption of one zone is about 2000 [m^3/h]. The similar field width has been assigned to temperatures. Number of slabs will be shown in the field of three characters wide. To the field that shows air/gas ratio the four-character wide field with two fractional decimal digits has been assigned. Period will be shown as the starting and finishing time separated by a hyphen.

For all the variables, apart from number of slabs the average values per hour are calculated (gas consumption is stored in [m^3/h]. The counter of slabs is archived. Based on its content the difference between beginning and the end of the interval (by means of the **diff** evaluator). Definition of that report is shown on the listing above. The example report is presented below.

THE ŁABĘDY STEELWORKS Pusher furnace
Operator: Marek Jabłoński Date: 20-06-1992

from-to	g1[m3]	g2[m3]	g3[m3]	p/g1	p/g2	p/g3	t1[°C]	t2[°C]	t3[°C]	tk[°C]	lk
22-23	379	1079	536	6,23	7,48	5,60	1269	1274	1281	1206	34
23- 0	947	1429	371	5,59	7,46	5,58	1266	1261	1269	1218	84
0- 1	478	952	561	6,24	7,52	5,57	1268	1258	1259	1179	4
1- 2	158	546	701	8,34	7,50	5,57	1268	1252	1257	1157	0
2- 3	141	716	692	8,36	7,46	5,59	1278	1260	1272	1178	30
3- 4	915	1564	373	5,55	7,22	5,60	1265	1274	1277	1220	48
4- 5	1187	1618	338	5,57	7,25	5,58	1270	1274	1275	1220	67
5- 6	1113	1597	455	5,62	7,16	5,56	1259	1275	1271	1206	46
6- 7	498	1244	486	5,55	7,48	5,59	1273	1274	1276	1203	0
7- 8	929	1419	389	5,60	7,36	5,57	1276	1270	1274	1212	5
8- 9	593	1018	444	5,60	7,48	5,59	1274	1259	1276	1201	0
9-10	712	1228	415	5,57	7,49	5,58	1259	1255	1267	1198	0
10-11	729	1229	415	5,59	7,49	5,61	1259	1255	1265	1195	15
11-12	694	1301	397	5,58	7,49	5,60	1262	1260	1266	1197	0
12-13	991	1533	326	5,56	7,20	5,60	1258	1269	1266	1210	7
13-14	673	1228	631	5,77	7,38	5,56	1244	1257	1266	1181	0
14-15	213	727	691	7,10	7,49	5,61	1278	1259	1272	1179	7
15-16	983	1541	352	5,58	7,27	5,57	1272	1258	1271	1216	26
16-17	884	1346	452	5,61	7,49	5,59	1276	1260	1270	1185	21
17-18	412	1024	512	5,58	7,48	5,61	1274	1268	1272	1191	28
18-19	1014	1409	389	5,61	7,47	5,58	1276	1262	1269	1204	6
19-20	754	1256	431	5,57	7,45	5,59	1270	1257	1267	1195	8
20-21	773	1187	417	5,58	7,49	5,58	1274	1254	1266	1195	5
21-22	947	1369	414	5,60	7,51	5,58	1277	1256	1265	1197	12

13.3. ASTEL - the Language for Computing the Process Data

For processing of both archived (mainly) and current process data the special language called ASTEL has been developed. The language enables calculation of arithmetical expressions that contain process data. Calculation can be executed in interactive mode. The expressions written in the ASTEL language can also be used for definition of the reports by means of the ASTER language that is going to be further discussed.

13.3.1. Components of the Language

Numbers
Variables
Moments
Operators
Functions
Parenthesis
Commas
Evaluators
Time Evaluators
Evaluators that Operate with Archived Data
Evaluators that Shift Beginning and Finish Moments
Evaluators that Operate with Archived Data with the Step Defined
Evaluators that Operate with Current Data
Alias Evaluator
Time as an Evaluator's Argument
Moments
Expressions that Include Evaluators
Expressions that Contain Functions
Expression that Contains Operators

The space character is used as a separator of components of expressions. Components of expressions must be separated with spaces only there, where missing of them would lead to ambiguous syntax understanding. The length of a single expression is limited to 256 characters. Both capital (upper case) and minor (lower case) letters are considered as the equivalent ones (*case-insensitivity*) Exception will be separately discussed. There are the following components of the ASTEL language:

- numbers;
- variables;
- moments;
- operators;
- functions;
- evaluators;
- parenthesis (brackets);
- commas.

Numbers

The number is any string that includes only digits and decimal point. Both integer and fractions are allowed (apart from the scientific notation). In case of decimal fractions the integer part or the fractional one are allowed to be empty.

EXAMPLES

```
1
1.2345
33.
.123
.
```

Variables

The variables are any strings that consist of letters, digits and underscores that begin with a letter. Number of characters that compose individual variables is not limited, although creation of the too long names can lead to the situation while there will be not enough memory and translation will not be executed. The names of variables can't be identical as the names of functions and evaluators. Any numerical value can be assigned to the variable, usage of the name of variable causes operation on the assigned value. The values should be assigned to the variables before they will be used in expressions (the

exceptions are variables that are arguments of evaluators and variables that appear on the left side in the assignment instructions).

EXAMPLES

x
s2
Reactive_Power

There are three pre-defined variables in the ASTEL language: **btime, etime** and **dtime** that denote the initial and terminating moments and the difference of them. The listed variables can appear only as arguments of the **time, year, month, day, hour, minute** and **second** evaluators.

The names of variables that are variables of the **asix** language may sometimes not meet requirements of the ASTEL language (they can include, for instance the characters '+', '-', '/', '=', etc.). Such variables can only appear as arguments of the evaluators. To provide possibility to execute lexical analysis the mentioned names should be bordered by "{}" curly braces.

EXAMPLES

{s=x+y/5}
{/////////}

Moments

Moments are specially created strings of characters interpretation of them enables date and time determining. Evaluation of a certain class of expressions of the ASTEL language requires defining (sometimes in indirect way) the beginning and finishing moment. The absolute and relative moments are distinguished. The absolute moments consist of numbers that denote year, month, day, hour, minute and second separated by backslash '\'. The number that denotes year can be shown in the abbreviated form - as the two last digits. Whether hours, minutes and seconds are omitted, the zero values are assigned. If the day number is omitted, the first day of the month is assigned. If the month number is omitted, the January is assigned. The strings that contain incorrect number values are not considered as moments (e.g. 13 at the month's position).

EXAMPLE

The absolute moments are the following strings of characters:

2001\6\3\12\20\10	3-rd of June	2001	12:20:10
03\1\1	1-st of January	2003	00:00:00
2002\12\3\18	3-rd of December	2002	18:00:00

Some absolute moments can be defined by use of the pre-defined strings of characters, that denote the beginning or end of some characteristic periods (of the year, month, day and week). These moments are defined as referred to the date of making the report. They are listed below (with a comment assuming, that the report was made on Tuesday, the 9th of March 1993 at 3.30 p.m.).

year>	1-st	January 1993	00:00:00
month>	1-st	March 1993	00:00:00
day>	9-th	March 1993	00:00:00
hour>	9-th	March 1993	15:00:00
week>	8-th	March 1993	00:00:00
year<	1-st	January 1994	00:00:00
month<	1-st	April 1993	00:00:00
day<	10-th	March 1993	00:00:00
hour<	15-th	March 1993	16:00:00
week<	15-th	March 1993	00:00:00
year|	1-st	January 1992	00:00:00
month|	1-st	February1993	00:00:00
day|	8-th	March 1993	00:00:00
week|	1-st	March 1993	00:00:00

The absolute moments defined in the above manner can be additionally shifted. Whether the numbers that denote hour, minute and seconds separated by the backslash '\' character occur just after the absolute moment standing for beginning of the period (e.g. **day>**), the notation denotes shifting the moment forward to the current one within the defined time interval. Whether such a string occurs just after the absolute moment standing for end of the period (e.g. **day<**), the notation denotes shifting the moment backward from the current one within the defined time interval.

Let's assume, that the report is being made Tuesday, the 9th of March 1993 at 3.30 p.m. Then:

day>5	9-th	March 1993	05:00:00
day<1\30	9-th	March 1993	22:30:00

Defining of the current moment in the simplified way is also possible. The '**&**' character is used for this purpose. Whether the numbers that denote hour, minute and seconds separated by the backslash '\' character occur just after '**&**' character, it denotes shifting the moment backward from the current moment within the defined time interval. Shifting of the moment forward from the current moment has not been defined (it would denote shifting to the future). The possibility of specifying of the moment related to the current time value can be especially useful for some types of reports.
Let's assume, that the report is being made Tuesday, the 9th of March 1993 at 11.20 a.m.

&	9-th	March 1993	11:20:00
&2\10	9-th	March 1993	9:10:00

Beside of absolute moments, as described above, the ASTEL language allows defining moments by indicating a month name. They determine beginning of this month or the beginning of a preceding month. Abbreviations are 3-letters and end with a character „>" or „|". In the first case such an abbreviation means the beginning of a given month and in the second case it means the beginning of its antecedent. When using them one should remember that the year of a given moment results from the date indicated for counting a report or from a shift of default moment.

Names of moments, which determine months, may be given in English or Polish. These names are as follows:

JAN| , FEB| , MAR| , APR| , MAY| , JUN| , JUL| , AUG| , SEP| , OCT| , NOV| , DEC|

EXAMPLES

```
Avg(power,JAN>,FEB>)      // average power for the January
Avg(power,DEC>,YEAR<)     // average power for the December
```

This mechanism is intended to use in reports based on a „symbol" instruction.

Moments of a period beginning and end allow determining the beginning and end of time period for ASTEL language evaluators. The beginning moment has a symbol "|" whereas the final one - "_". These moments are used only in month reports where it is not known how much one should move with respect to a default moment, e.g. to the beginning, in order to be in a specified distance from the end (because months have different length. The syntax is the same as for the moment, now "&".

EXAMPLES (PERIOD IS A MONTH)

```
Avg(power,|,|240)         // average power for the first decade
Avg(power,|480,_)         // average power for the last decade
```

Beside the absolute moments the relative ones can also be used. They are defined with regard to the previously defined moments (as well in indirect way). Strings of characters that denote relative moments consist of numbers that stand for hour, minute and second separated by the backslash '\' character. Such a string should be preceded by the character '>' (greater than) or '<' (less than) that determined direction of shifting of time (forward or backward respectively). Values of minutes and seconds greater than 59 are considered as invalid. The number of hours greater than 24 is allowed for relative moments - this denotes multi-days shifting. Whether the values of minutes and seconds are omitted, the zero values will be assigned. There are some examples below. Whether the absolute moment is the 7th of April 1993, 12:00:00, then:

>48	9-th	April	1993	12:00:00
>1\30	7-th	April	1993	13:30:00
<0\0\30	7-th	April	1993	11:59:30

Operators

Operators are the lexical elements that stand for individual operations. There are allowed both binary and unary operators. The expressions are arguments for operators. Every operator has its own priority that defines the order of evaluation of expressions. The list of operators is shown below.

+	(plus)	addition
-	(minus)	subtraction
*****	(asterix)	multiplication
/	(slash)	division
^	(arrow)	involution
++	(double plus)	maximum
--	(double minus)	minimum
||	(double stroke)	logical sum (or)
&&	(double ampersand)	logical product (and)

^^	(double arrow)	exclusive or (xor)
!	(exclamation)	logical negation (not)
==	(double equal)	equal
<>	(less-greater)	not equal
>>	(double greater)	greater than
<<	(double less)	less than
>=	(greater-equal)	greater or equal
<=	(less-equal)	less or equal
?	(question mark)	round
=	(equal)	assignment
-	(minus)	sign change

EXAMPLE

Expression that contains operators:

```
(x+y)*z+1.3/(x-2)
p=-3.33*5
2++99--(-11)--0
z=x=3
```

The relational operators are the following:

==	binary operator of equivalence (equal)
<>	binary operator of non-equivalence (not equal)
>>	binary operator "greater than"
<<	binary operator "less than"
>=	binary operator "greater than or equal to"
<=	binary operator "less than or equal to"

The operators of the above group are of *True* (1) value whether the condition is met or *False* (0) value otherwise, e.g. "a==5" is *True* if value of the *a* variable is 5 at the moment of comparison. Multiple applying the same operators is also allowed in one expression, but be careful. However the expression:

5>>4	is *True*
5>>4>>3	is *False*, because operations are executed from left to right, so (5>>4)>>3 gives (1>>3), what is 0 (*False*).

The logical operators are the following:

!	unary operator of logical negation (NOT)
||	binary operator of logical sum (OR)
&&	binary operator of logical product (AND)
^^	binary operator of exclusive or (XOR)

The operators of the above group can be True (1) or *False* (0). **NOTE:** They are logical, not bitwise operators, so they operate with the whole words. It is assumed, that the value of the operand different from zero denotes *True*, and equal to zero denotes *False*.

The minimum/maximum operators are the following:

++	binary operator of maximum
--	binary operator of minimum

The value of greater (maximum) or less (minimum) of the two arguments is assigned as the result of operation for the above group. So as the value of the expression:

2++8	is 8, but
2++99--(-11)--0	is -11

Implementation of such operators provides large abilities, but one should be careful with existing possibilities of creation of expressions that look rather sophistically, e.g. the expression:

a>>3||x+3++6 is correct

Above-mentioned operators provide possibilities of conditional calculation.

Functors

The functors are represented by string of characters that stand for unary mathematical operation. From the point of view of evaluation such expression are similar to unary operators. An expression serves as an argument of every function. Whether a number or a variable is used as an argument of a functor, the parenthesis where the expression is written can be omitted. The functors, just as operators, own the assigned priority.

sin	sinus
cos	cosinus
tg	tangens
arcsin	arc sinus
arccos	arc cosinus
arctg	arcus tangens
sinh	hyperbolic sinus
cosh	hyperbolic cosinus
tgh	hyperbolic tangens
abs	absolute (also brackets **[]**)
exp	exponent
ln	natural logarithm
sqr	square
sqrt	square root (also the √ character)
rnd	random number
ent	integer part of a number
frac	fractional part of a number

EXAMPLE

Expressions that contain functors:

```
sin x
sin(x)+cos(y^2)
sin ln [x*2.15]
sin (z=x=3)
```

The following expression are equivalent each other

sin x	sin(x)
[x+2]	abs(x+2)
sqrt 3	sqrt(3)

Evaluators

The evaluators are represented by string of characters that denote execution of a certain function with several arguments (or a function of several variables). Number of arguments and type of them depend on the type of the evaluator. The numerical values are not the only allowed arguments of evaluators, they can also be variables and moments. Individual arguments are closed in parenthesis after a name of an evaluator and they are separated by means of spaces or commas. The last arguments from the end of the list can be omitted,

then default values are assigned instead. The first argument cannot be omitted, but the enclosing parenthesis can be abandoned whether the only one argument is used. If part of arguments are default ones that precede the list of explicitly declared arguments, the character **@** is to be used to denote default arguments. Because of number and type of arguments, all the functions have been subdivided into groups.

Evaluators that Operate with Archived Data

val	value of the variable at the moment
int	sum of values in the interval
diff	difference of values at the end of the interval
max	maximum value
min	minimum value
grad	maximum increase of the value
avg	average value (breaks in archives are considered as zeroes)
acc	average accumulated value (breaks in archives are omitted). Evaluators **avg** and **acc** consider in different way the areas where data, because of lack of archived data, were missed. Therefore, whether during the discussed period the variable was constantly 100 during 50% of time and data were missed for the rest of the period, the **avg** evaluator will indicate 50, but the **acc** indicator will adopt the value of 100.
inc	sum of increments
spos	like **inc**, but after an occurrence of the value decrement, the value of the first measurement after the decrement is added
sposh	like **spos**, but increments of two values are also added although a hole occurred between them *Example:* Content of the archive: 2,3,4,3,(hole),5 Result: (3-2)+(4-3)+3+(5-3) = 7
dec	sum of decrements
sneg	like **dec**, but after an occurrence of a value increment the value of the first measurement after the increment is added
snegh	like **sneg** , but negative increments are also added although a hole occurred between them. *Example***:** Content of the archive: 7,4,3,5,4,(hole),2 Result: (4-7)+(3-4)+5+(4-5)+(2-4) = -2
var	variance
count	number of values within the interval
inf	the ratio of time, while measurements were available to the whole time that refers to the discussed interval. Value of the **inf** evaluator can vary from 0 to 1. The value of 1 means, that the system operated with no errors, breaks in archives and turning off. Value of 0

indicates no valid measurements within the interval.

Evaluators that operate with archive data are called by means of the following sequence

evaluator_name(name,begin,end,min,max,type)

where:

name - name of the process variable of the **asix** system;
begin - beginning moment;
end - finish moment;
min - minimum allowed value of the variable (by default the - minimum number of *float* type);
max - maximum allowed value of the variable (by default the maximum number of *float* type);
type - type of the process variable archive (by default **m**).

Name and **type** constitute description of the process variable created for needs of the archiving program ASPAD. Declaration of name is obligatory, other arguments can be omitted, and consequently the default values will be assigned. The **begin** and **end** parameters define the period referred by the calculation. Parameters **min** and **max** enable limitation of range of variables to reasonable values. When missed, the maximum range allowed for floating-point numbers will be adopted. For the **val** evaluator the finish moment is unimportant. The letters **B, M** and **H** can be used as values of the **type** parameter. They denote respectively

B searching for archive data in B-type files (relational database Paradox),
M searching for archive data in M-type files,
H searching for archive data in H-type files,
D searching for archive data in D-type files,
Y searching for archive data in Y-type files.

Whether the lowercase letters **b, m, h,d** and **y** are quoted here, the archive of indicated type will be tried to be open and, if not exists, the attempt to open the archive for the given variable of **B, M, H, D** and **Y** type consequently will be carried out**.**

EXAMPLES

avg (power,93\1\3,93\4\8)- denotes calculation of average value for the process variable called **POWER** with the **m** type of archives for the time interval from the 3rd of January to the 8th of April 1993.

avg (power,@,@,0,50) **-** denotes calculation of average value for the process variable called **POWER** with the **m** type of archives for the default time interval while the value range is limited to values from 0 to 50.

Evaluators that Operate with Archived Data with the Step Defined

sval value of the variable at the moment
sint sum of values in the interval

smax	maximum value
smin	minimum value
savg	average value
scount	number of values within the interval

Evaluators, which operate with archived data, are called by means of the following sequence:

evaluator_name (name, step ,begin, end, min, max, type)

where:

name	- name of the process variable of the **asix** system;
step	- step, in seconds, applied for sampling of the timing (60 by default);
begin	- beginning moment;
end	- finish moment;
min	- minimum allowed value of the variable (by default the minimum number of *float* type);
max	- maximum allowed value of the variable (by default the maximum number of *float* type);
type	- type of the process variable archive (by default **m**).

With evaluators operating on archives with a specified step the **stype** function is connected. It determines a way of interpolation. The STYPE function has one argument, which is an integer number 0,1,2,3. They mean respectively:

0 – selection of the nearest measurement;
1 – selection of linear interpolation (by default);
2 – selection of the previous measurement;
3 – selection of the next measurement.

The evaluators listed in this group are very similar to the ones described in the previous group. The only difference is, that for the previous group **all** the measurements related to the given interval and registered in the archives have been taken for calculation, but the evaluators described in this group sample the registered timing at the established step and exclusively those measurements are further processed.

The difference is rather significant, let's assume that the maximum value within the one-hour long interval is going to be found and, for instance, the step of 5 minutes is applied. As a result of sampling, instead of the really maximum value that occurred at 1:38, the slightly lower value related to 1:40 will be found.

The purpose of individual parameters is the same as for evaluators of the first group. The **step** parameter denotes step for sampling the archives. Applying of very little steps for large intervals is not recommended, because it leads to prolongation of calculation time. The finish moment is unimportant for the **sval** evaluator.

EXAMPLE

savg (power,3600,93\1\3,93\4\8) - denotes calculation of average value for the process variable called **POWER** with the **m** type of archives for the time interval from the 3rd of January to the 8th of April 1993 in such a manner, that values for

calculation of the average are sampled with the step of one hour.

Time Evaluators

time	time (number of seconds passed from the 1st of January 1980)
year	year
month	month
day	day
hour	hour
minute	minute
second	second
tmax	determines a time, for which a maximum occurred
tmin	determines a time moment, for which a minimum occurred

Time evaluators are called by means of the following sequence:

evaluator_name (time)

where:

time - name of the beginning moment BTIME, finish moment ETIME or difference of them DTIME**,** it can be either absolute or relative moment.

EXAMPLE

hour (etime)	the hour that refers to the (default) finish moment
hour (&)	the hour that refers to the current moment
month (month>)	the month that refers to the begin of current month

Evaluators that Shift Beginning and Finish Moments

over>	start of exceeding the upper limit;
over<	end of exceeding the upper limit;
under>	start of exceeding the lower limit;
under<	end of exceeding the lower limit;
limit	the ratio of period, while the value of the variable exceeded the established limit to the total duration of time referred to the considered interval (does not shift moments - falls into the range from 0 to 1);
nlimit	the ratio of period, for which the variable value was smaller than the established limit to the period referred to the considered interval (does not shift moments – assumes the values from the range from 0 to 1);
tmax>	shifts the beginning moment to the time of maximum occurrence;
tmin>	shifts the beginning moment to the time of minimum occurrence.

Evaluators, that shift beginning and finish moments, are called by means of the following sequence:

evaluator_name (name, limit, begin, end, type)

where:

name	- name of the process variable;
limit	- constant or variable defining the limit (0 by default);
begin	- beginning moment;
end	- finish moment;
type	- type of the process variable archive (by default **m**).

Declaration of name is obligatory, other arguments can be omitted, and consequently the default values will be assigned. The value of evaluators of this group is always 0 or 1. The value of 1 denotes that the point, where the tested variable exceeded the upper limit value (or lower limit value) was found within the considered interval. The side effect is shifting of the beginning or finish moment to the found point. The **limit** evaluator is an exception. Its value falls into the interval from 0 to 1 and it does not cause shifting of the moments. Values of the **type** parameter can be the same as for evaluators of the first group. The graphical interpretation of exceeding of the upper (or lower) limit is shown on the picture.

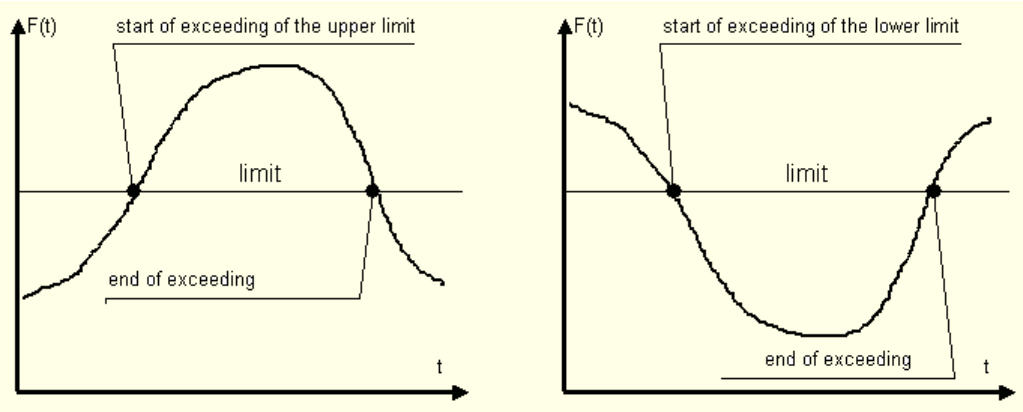


Figure 13-10. The Graphical Presentation of Start and End of Exceeding.

EXAMPLES

over> (power,100)	- denotes searching for the moment, when the process variable **POWER** of the **m** type of archive will exceed, within the considered interval of timing, the upper limit of 100. Whether such a moment is found, the value of the evaluator will be 1 and the found moment will become the beginning one.
limit (power,100)	- denotes definition for the process variable **POWER** of the **m** type of archive, what is the contribution of measurements of values over 100 within the considered interval of timing.
m = max(**z**)	- variable **m** contains the maximum of variable **z.**
t = tmax(**z**)	- variable **t** contains the time when the maximum occurred.

v = val(**x**,**t**) - variable **v** contains the value of variable **x** for the time **t**.

Further constructions of the type as below are possible:

t = **t**+3600 - increment of the time t of 1 hour.

v = val(**x**,**t**) - **v** contains the value of variable **x** for the new time **t**.

If operators of the **tmax>** and **tmin>** group are used then the first example, given above with a use of these evaluators, may be written as follows:

m = max(**z**) - variable **m** contains the maximum of variable **z**.

t = tmax>(**z**) - variable **t** contains the time when the maximum occurred.

v = val(**x**) - variable **v** contains the value of variable **x** for the time **t**
but it should be remembered that the default beginning is
shifted for each next line (what may be intentional by the way).

Evaluators that Operate with Current Data

read **-** current value of the process variable is read from a controller.

get - current value of the process variable is got from the system tables.

This evaluator is called by means of the following sequence:

evaluator_name (name, min, max, mask)

where:

name - name of the process variable of the **asix** system;

min - minimum allowed value of the variable (by default the minimum number of *float* type);

max - maximum allowed value of the variable (by default the maximum number of *float* type);

mask - optional mask for the selected bits of the word (by default no masking).

The use of the **get** evaluator instead of the **read** evaluator is always recommended, if direct reading of the variable from the controller is not necessary. This method significantly increases speed of system operation.

EXAMPLE

read(x,@,@,0x00FF) - denotes reading of value of the *x* variable and executing the bitwise product with the hexadecimal mask 0x00FF before conversion to the *float* value.

NOTE *So as the process variable can be either 16-bit or 32-bit fixed-point number, or floating-point number, the 16-bit or 32-bit mask is to be respectively used. Applying the 32-bit mask for floating-point number requires some knowledge about floating-point format. Applying the 32-bit mask for the 16-bit variable leads to removing the more significant word of the mask. Applying the 16-bit mask for the 32-bit variable leads to completion of the mask with zero-value more significant word.*

Implementation of masking technique enables introducing the new virtual variables, being "cut-off" from pieces of various other variables.

EXAMPLES

read(x,@,@,0xFF) **-** cutting off the less significant byte;
read(x,@,@,0xFF)/256 - cutting off the more significant byte;
read(x,@,@,0x80)/2+read(y,@,@,0x3) - creation of the new variable by means of cutting off the two least significant bits of the **Y** word and adding to them the bit of the weight of 2^7 of the **X** word, shifted left by position.

Applying of logical expression provides possibility to introduce a mechanism that replaces the ***if*** instruction. For instance, the following piece of a program that contains the ***if*** instruction

```
if ((a>30)||(b&0x10))          // where a and b - process variables
  w=50
else
  w=0
```

is to be written in the ASTEL language as

```
a=read(a)
b=read(b,@,@,0x10)
w=((a>>30)||b)*50
```

Alias Evaluator

alias - the value of *alias* declared in the initialization file.

This evaluator is called by means of the following sequence:

evaluator_name (name)

where:

name - name of the alias. The **alias** evaluator is used, if some values used in the expression are to be read from the initialization file. This method is reasonable whether many similar applications are created that differ only slightly in details (e.g. parameter values for certain similar expressions).

EXAMPLE

Expressions that include evaluators:

max d2
max d2 - min d2
int(power)/count(power)*0.8
max(steam,91\6\6,91\6\7)
var(z,week>,week<)
1.2*read(x)+200

Time as an Evaluator's Argument

It is possibility to operate with the time as an argument of evaluators of the group **day, hour, second** etc.... . E.g. the constructions are possible:

t = time(&)	variable **t** contains the actual time;
h = hour(&)	variable **h** contains an actual hour e.g. 15;
t = t+3600	increment of the time t of 1 hour;
s = time(**t**)	variable **s** contains the actual time increased with 1 hour;
g = hour(**t**)	variable **g** contains an actual hour + 1 e.g. 16.

Parenthesis (Brackets)

Both parenthesis "**( )**", used for defining the order of executing individual operations that constitute the expression, and square brackets "**[ ]**", for calculation the absolute value, are allowed to be used.
The curly braces "**{ }**" can be used as delimiters of variables of the **asix** system that are used as the first parameters of some evaluators.

Commas

Characters of commas '**,**' are used as separators (interchangeable with the spaces) due to separate individual arguments of the evaluator. The following examples contain equivalent expressions:

EXAMPLES

count(X,93\3\1,93\3\3)
count(X 93\3\1 93\3\3)

13.3.2. Grammar of the ASTEL Language

<digit>	::=	**0**|**1**|**2**|**3**|**4**|**5**|**6**|**7**|**8**|**9**
<letter>	::=	**A**|**B**|**C**|**D**|**E**|**F**|**G**|**H**|**I**|**J**|**K**|**L**|**M**|**N**|**O**|**P**|**Q**|**R**|**S**|**T**|**U**|**V**|**W**|**X**|**Y**|**Z**
<integer>	::=	<digit>|<integer><digit>
<number>	::=	<integer>**.**<integer>|<integer>**.**|**.**<integer>|**.**
<variable>	::=	<letter>|<variable><letter>|<variable><digit>|<variable>**_**

<character> ::= *any character entered from the keyboard apart from {}*

<name> ::=
{<character>**}**|**{**<name><character>**}**

<second> ::= <integer>[0-59]

<minute> ::= <integer>[0-59]

<hour> ::= <integer>[0-24]

<day> ::= <integer>[1-31]

<month> ::= <integer>[1-12]

<year> ::= <integer>[1980-2047]

<shift> ::= **>**|**<**

<shift_value> ::= <integer>|<integer>****<minute>|
<integer>****<minute>****<second>

<period> ::=
year>|**month>**|**day>**|**hour>**|**week>**|
jan>|**feb>**|**mar>**|**apr>**|**may>**|**jun>**|**jul>**|
aug>|**sep>**|**oct>**|**nov>**|**dec>**|
jan||feb||mar||apr||may||jun||jul||aug||

sep||oct||nov||dec||year||month||day||hour||week||

year<|**month<**|**day<**|**hour<**|**week<**

<moment> ::= <year>****<month>****<day>|
<year>****<month>****<day>****<hour>|

<year>****<month>****<day>****<hour>****<minute>|

<year>****<month>****<day>****<hour>****<minute>****<second>|

<period>|<period><shift_value>|<shift><shift_value >|**&**|**&**< shift_value >

<operator1> ::= **-**|**~**|**!**

<operator2> ::= **+**|**-**|*****|**/**|**^**|**=**|**++**|**--**|**||**|**&&**|**^^**|**==**|**<>**|**>>**|**<<**|**>=**|**<=**|?

<funktor> ::=
sin|cos|tg|arcsin|arccos|arctg|sinh|cosh|tgh|

abs|exp|ln|sqr|sqrt||rnd|ent|frac

<evaluator> **::=**
val|int|diff|max|min|grad|avg|acc|var|count|inf|inc|dec|spos|sneg|

sposh|snegh|sval|sint|smax|smin|savg|scount|tmax|tmin|

tmax>|tmin>|time|year|month|day|hour|minute|second|

limit|over>|over<|under>|under<|read|get|alias

<separator> ::= **,**|<space>

<component>::= <number>|<variable>|<moment>|<name>

```
<list>              ::=
                        <component><separator><component>|
                              <list><separator><component>
<argument>          ::=       <number>|<variable>|
                              (<expression>)|[<expression>]|
                              <functor><expression>|

                        <evaluator><component>|<evaluator>(<li
                        st>)
<expression>::=     <argument>|
                              <operator1><expression>|

                        <expression><operator2><argument>
```

The grammar of the ASTEL language has been written by means of the BNF notation. The <character> rule should be interpreted as any character, which can be entered from the keyboard, with exception of "{}" curly braces. The extension to number specification, which consists in placing its allowed range in square brackets, has been used in some rules.

13.3.3. Evaluation of Expressions

The method of evaluation is subjected to the following rules:

- the order of execution of individual operation that creates an expression is established by priorities.
- the order of execution of operations, which is result of priorities, can be changed by means of parenthesis.
- whether the priorities are the same. the operations are executed from left to right.
- functions are executed from right to left.
- assignment operations are executed from right to left.

The following table contains list of operators and functors together with priorities assigned to them.

Table 13-1. The List of Operators and Functors for the ASTEL Language.

Operation	**Priority**
Function	11
Involution	10
Change of sign	9
Negation	8
Rounding	7
Multiplication	6
Division	6
Addition	5
Subtraction	5
Maximum	4
Minimum	4
Logical AND	3
Logical OR	3
Logical XOR	3
Equivalence	2
Non - equivalence	2
Greater than	2
Less than	2
Greater than or equal to	2
Less than or equal to	2
Assignment	1

EXAMPLES OF EXPRESSIONS

Table 13-2. Examples of Expressions for the ASTEL Language.

Expression	Evaluated as	Result
2+3*4	(2+(3*4))	14
(2+3)*4	((2+3)*4)	20
4-1-1	((4-1)-1)	2
4-(1-1)	(4-(1-1))	4
2*2^3	(2*(2^3))	16
(2*2)^3	((2*2)^3)	64
-2^4	(-(2^4))	-16
(-2)^4	((-2)^4)	16
√2	((2))	1.4142
ln 0	(ln(0))	error
sin 2^2	((sin(2))^2)	0.8268
sin(2^2)	((sin(2^2)))	-0.7568
sin ln 1	(sin(ln(1)))	0
x=y=3	(x=(y=3))	3
k=k+1	(k=(k+1))	3 if *k* was before equal to 2
i=(i=1)+4	(i=((i=1)+4)	5
(z=z+1)+(z=z+2)	((z=z+1))+(z=z+2)))	6 if *z* was before equal to 1
(i=k)+4	((i=k)+4)	5 if *k* was before equal to 1
avg x	(avg(x))	average value of *x* eg. 100
sint x/scount x	((sint (x)/scount(x)))	average value of *x* (calculated in another way)
avg(x,91\6\5,91\6\6)	(avg(x,91\6\5,91\6\6))	As above (with the list of arguments)
hour btime	(hour (btime))	e.g. 15
over>(x,100)	(over>(x,100))	0 or 1
get(x,0,100)	(get(x,0,100))	Current value of variable (from 0 to 100)
5>>4>>3	(5>>4)>>3	0 (false)
2++99--(-11)--0	(((2++99)--(-11))--0)	-11
1>>3||2+3++6	(1>>(3||((2+3)++6)))	0 (false)
pi=2*arctg(1/0)	(pi=(2*(arctg(1/0))))	3.1415 (Note: division by 0)

13.3.4. Diagnostics

SYNTAX ERROR
TOO COMPLICATED EXPRESSION
NON-DEFINED VARIABLE
RUN-TIME ERROR
EVALUATION ERROR

CALCULATION ERRORS
DOMAIN ERRORS
ARCHIVES ERRORS

COMMUNICATION ERRORS
FORMAT ERRORS
SYNTAX ERRORS
OTHER RUN-TIME ERRORS
ERRORS IN DIALOG MODE OF OPERATIN

ERRORS AT CALCULATION OF REPORTS
TYPES OF ERRORS

A number of errors may occur during translation and evaluation of expression in the ASTEL language. There can be syntax errors, semantic errors and errors of evaluation (run-time). Whether errors occur at the stage of translation of the expression, the expression is not evaluated. The errors that may occur apart from the evaluation process have been subdivided into:

- **critical** (fatal) errors that cause interruption of evaluation (e.g. reference to a non-existing variable),
- **conditional critical** errors, after having occurred of them the evaluation process may go on,
- **non-critical** errors, they do not cause interruption of evaluation (e.g. holes in archive).

Depending on the mode of use of the ASTEL language the above errors are alerted in various manner. The more complete diagnostics is available if the ASTEL language is used in interactive mode (by means of so called DIALOGS). In this mode the expressions are analyzed and evaluated on-line, then the calculated value or error massage (and possible warnings) are displayed. In the case of calculation of reports the erroneous situations will be alerted in the simplified way.

Errors in Dialog Mode of Operation

SYNTAX ERROR

The syntax error occurs whether syntax of the entered expression does not conform to the grammar rules of the ASTEL language. In this case the message **syntax error** that precedes the expression, where the error occurred, will be outputted. The individual (lexical) symbols of that expression will be separated by spaces and, additionally, the '**?**' character will be placed just before the first incorrect or unrecognizable symbol. Whether the syntax error that consist in various number of open and closed parenthesis occurs, the message **number of parenthesis** will be displayed (it is not known whether parenthesis are missed or there are too many of them). Evaluation is not executed after syntax error.

EXAMPLES

2+++3
syntax error 2 ++ ? + 3
-S syntax error or too complex

ln+2
syntax error LN ? + 2
-S syntax error or too complex

(2+3
syntax error – number of parenthesis
-S syntax error or too complex

TOO COMPLICATED EXPRESSION

This error occurs, whether the entered expression consists of more than 255 lexical components. The expression should be simplified or replaced by greater number of simpler ones. This error never occurs in the dialog mode, because entering of the longer line is impossible. When occurs in the report calculation mode it causes output of diagnostic message and disability of report output.

NON-DEFINED VARIABLE

This error occurs, whether there was used the variable that had not been defined before (did not appear on the left side of the substitution). It is not a syntax error, but a semantic one (occurrence of it is tested during the evaluation process). Whether occurs, the missed variable has to be

defined, then evaluation is to be executed again (or write correctly the name of the previously defined variable).

EXAMPLE

z+3
variable not defined
-S syntax error or too complex

z=2
no errors
2
z+3
no errors
5

RUN-TIME ERROR

This error occurs, whether the evaluation process cannot be completed because of any reason, or, after having the process completed the stack overflow took place. Instead of the result of evaluation the message **run-time error** is displayed.

EVALUATION ERROR

The evaluation error occurs, whether the evaluation process is completed, but any critical (fatal) errors or conditional critical (fatal) errors occurred, because of them the result of evaluation became not-a-number. For instance, not-a-number is the result of division by zero (infinity). Not-a-number can be used for further calculation provided that the result of them will be a correct value. In case of evaluation error the massage **evaluation error** will be displayed.

EXAMPLES

ln 0
evaluation error
-D nonpositive arg of logarithm

1/0
runtime error
-C division by zero
-C overflow

arctg(1/0)
-C division by zero
-C precision lost

Whether during the evaluation process non-critical errors or conditional critical (fatal) errors occur, they will not cause the evaluation error. The result of calculation is outputted, but it is calculated conditionally. Such a situation happens rather frequently. It may take place, for instance, while the average value is calculated and data collecting is interrupted (e.g. caused by turning a computer off). Instead of evaluation error and lack of result the average calculated for continuous intervals (breaks are ignored)

will appear. The conditionally calculated result can be checked by means of a command in the appropriate dialog window.

Types of Errors

After an error occurred the actual reason of it can be checked. Execution of "?" command will cause output of information that explains why the error occurred. So as there could be many reasons, information of them can occupy some consecutive lines. Every line starts with a marker that indicates falling of the error into the defined group of errors. Meaning of the markers is as follows:

-C	calculation errors,
-D	domain error,
-A	archives errors,
-V	communication errors,
-E	run-time errors,
-I	other errors,
-F	format errors,
-S	syntax errors.

CALCULATION ERRORS

-C invalid operation critical (fatal) error

The not allowed operation, e.g. 0/0 occurred, the not-a-number was used as an operand or stack pointer error of the co-processor took place. Calculation halted.

-C mantissa denormalization non-critical error

The denormalization of the mantissa occurred (the mantissa is not normalized, i.e. it falls out of the 1/2 - 1 interval at minimum value of the exponent). The least significant digits of the operand are lost (the operand is close to zero value). Moreover, lost of precision will occur. Calculations go on.

-C division by 0 conditional critical (fatal) error

Division by zero occurred. The not-a-number standing for $+\infty$ or $-\infty$ will be adopted as a result. Calculations go on conditionally.

-C overflow conditional critical (fatal) error

The result of calculation exceeds the allowed range of numbers (too large absolute value of a number is obtained). The not-a-number standing for $+\infty$ or $-\infty$ will be adopted as a result. Calculations go on conditionally.

-C underflow non-critical error

The result of calculation is close to zero (too small absolute value of a number is obtained). Zero value will be adopted as a result and calculations go on. It leads to lose of precision.

-C precision lost non-critical error

Rounding operation was executed during calculation. The result is not exact. Calculations go on.

DOMAIN ERRORS

-D negative arg of square root conditional critical (fatal) error

The argument of the square root function is negative. Value of zero is adopted and calculations go on conditionally.

-D non-positive arg of logarithm conditional critical (fatal) error
The argument of the logarithm function is non-positive. The not-a-number is adopted and calculations go on conditionally.

-D arcus sin cos domain error conditional critical (fatal) error
The argument of the *arcsin* or *arccos* function does not fall into the interval from $-\pi$ to $+\pi$. The not-a-number is adopted and calculations go on conditionally.

-D too little population non-critical error
The number of measurements within the given interval is too little for executing calculations. Such a situation can occur e.g. at the attempt to calculate average value for an empty interval or calculation of difference between the lower and upper values at the ends of an interval but less than two measurements fall into it. The zero value is adopted and calculations go on.

-D invalid conversion critical (fatal) error
There occurred an error during conversion of a floating-point number to an integer one (integer out of range).

ARCHIVES ERRORS

-A channel open error critical (fatal) error
The ASPAD program cannot open a channel. Invalid specification of an archive variable or too many channels open at the time can be reasons of this error.

-A file seek error - no data critical (fatal) error
The error occurs while operation of positioning (searching for the beginning of data scanning area) is to be executed. Too far reference to the past can be the reason of this error.

-A reference to the future non-critical error
The reference to the future data. The zero value is adopted and calculations go on.

-A no data non-critical error
There are no archived data for the given period. The zero value is adopted and calculations go on.

-A later time non-critical error
There are no archived data for the given period (or outside of it). The zero value is adopted and calculations go on.

-A holes in archive non-critical error
Data acquisition within the checked period was interrupted. The zero value is adopted and calculations go on.

-A no measure critical (fatal) error
For the declared time (and in the neighborhood defined by limit conditions) there are no measurements. This error is able only for the **val** evaluator.

-A no data from a server critical (fatal) error
This error can occur only for network configuration of the program whether data cannot be transmitted by means of network. Break of network

connection, defect of transmission route or invalid setting of network parameters can be reasons of this error.

-A negative time increment noncritical (fatal) error
This error can occur only when next measure from archive has earlier time. Switching between winter and summer time and bad data from data sources can be reasons of this error. Calculations go on.

COMMUNICATION ERRORS

-V variable not ready non-critical error
The data have not been refreshed at the established time. The previously received value is adopted. Calculations go on.

-V variable not defined critical (fatal) error
The variable with the given name has not been defined yet. The correctness of written name should be checked.

-V variable not reliable non-critical error
Value of the variable is not reliable (e.g. it has not been refreshed for a long time). Calculations go on

-V transmission error critical (fatal) error
No transmission through communication routes. The process variable cannot be received.

FORMAT ERRORS

-F invalid format
The evaluation process has been completed correctly, but the result cannot be accordingly outputted following the established format (too little number of fields provided). This error cannot occur at the dialog mode (while format of the result is not specified), but only during calculation of the reports. The error is then alerted in the simplified way.

SYNTAX ERRORS

-S syntax error of too complex
The syntax error occurs if the expression was too complicated. Explanation of this error type was presented above.

OTHER RUN-TIME ERRORS

-E variable not defined critical (fatal) error
Referring to the non-defined variable occurred during evaluation.

-E invalid argument critical (fatal) error
Invalid argument of the operator occurred. Either type or range, or number of arguments may not conform to the rules. Correctness of arguments is checked during the evaluation process.

-I alias conversion error critical (fatal) error
The error of conversion of the text alias to the numerical value occurred at evaluation.

-I alias not found critical (fatal) error
The text alias that is called up by the alias function does not exist in the initialization file.

-E data simulation non-critical error
The program that calculates a report uses simulated data. The obtained results are practically useless.

Errors at Calculation of Reports

While calculation of reports large number of the ASTEL language expressions is evaluated. The result of every evaluation is a number that is to be inserted into the appropriate field. Types of errors are mainly the same as while working in the dialog mode. Information of the error that took place must only appear in the box with appropriate dimensions. All the syntax errors are detected on the stage of analysis of definition of a report, so they cannot occur during calculation of it. The only exception is the error of using a non-defined variable. While an error occurs, the appropriate alerting text is written instead of the result.

The text of description of the error is inserted in the place of the result. If the area for the result is wider than the error message needs, the message is centered within the area. If the area width is smaller than length of the text (four characters), the last characters of the message are cut out.
The format errors, that did not occur at dialog mode of operation are caused by specification of too little width of the field for the presented result (the number 1000 cannot be written by 3 digits - the string "?FR" will appear).

Table 13-3. The List of Error Types Typical For Report Generation.

Type of Error	**Description**
calculation errors	?CAL
domain errors	?DFN
format errors	?FRM
communication errors	?COM
archiving errors	?ARC
variable not defined	?VND
bad argument	?BAR

The results calculated conditionally, i.e. those, where non-critical errors occurred, are displayed with the '?' character on the end.

13.4. ASTER - the Language for Report Definition

The Aster language has been developed specially for definition both the content and the form of reports created within the **asix** system.

13.4.1. Components of the Language

Comments
Keywords
Formulas
Strings
Numbers
Styles
Markers
Lines
Special Characters

Definitions of reports composed in the ASTER language are consisted of a set of declarations and instructions. They can include the following components:

- comments,
- keywords,
- formulas,
- strings,
- numbers,
- styles,
- markers,
- lines,
- special characters,

The format of the report definition has not been defined in the strict manner. The individual components of the definition can be separated by means of any space, tabs and new line (CR+LF) characters. Apart from the **format** instruction that refers to the symbols that should be previously defined, the order of appearance of declaration is not important. Both capital (uppercase) and small (lowercase) letters are considered as the same ones (case insensitivity).

Comments

The comment is a text that explains definition of the report and is not analyzed by the language compiler. It is constituted of any string of characters included in the curly braces "{}". The comment can also start from the slash '/' and be as long as to the end of line.

EXAMPLE

```
{Comment}
{Comment
can even extent
across multiple lines}
/Comment
```

Keywords

Keywords are reserved strings of characters usually used for instructions and declarations. The 16 of keywords listed below are provided. The purpose of individual words will be further discussed.

name	**delimiter**	**marker**	**begin**
step	**repeat**	**count**	**symbol**
format	**header**	**record**	**input**
alias	**ctrl**	**export**	**text**
nowarning	**noerror**		

Formulas

Formulas are any expressions of the run-time data processing ASTEL included in the apostrophe characters. The syntax and purpose of them have been discussed in the chapter that has been devoted to the ASTEL language. Formulas may be calculated. As a result of evaluation either the numerical result or the error message is passed.

EXAMPLE

```
'1'
'arctg(1/0)*(3+2)'
'z=avg(POWER,day>)*0.8'
```

Strings

Strings are any consecutive sets of characters included in the quotation marks **"'"**. The string cannot contain the quotation mark itself. Strings are used to store data of text character.

EXAMPLE

```
"report"
"12.5"
""
```

Numbers

Numbers are any strings of digits. The ASTER language uses only natural numbers for report definition (the real number can be used as components of formulas as well). The numbers are used for definition of dates, time and the way of presentation of results.

EXAMPLE

```
0
123
```

Styles

Styles are the strings of characters defining the manner of location of results within the area provided for their presentation. The four styles are defined: **-L, -R, -C** and **-Z** that denote respectively: filling with spaces on the left or on the right, centering of the result within the area of presentation or filling with zeroes left of the number.

EXAMPLE

'1':5:0-C	/ denotes, that the string ‘1’ will be presented in the center of / the five-character wide field

There is a possibility to modify standard formats denoted with letters **L**, **R**, **C** and **Z** by adding declarations included in the group of characters controlling the printout of warnings. After a character denoting a style i.e. **L**, **R**, **C**, **Z,** characters controlling the printout may be placed in case of an error or warning of a specified type. Following characters are allowed:
- **F** – reference to the future,
- **H** – a hole occurred,
- **B** – time backout occurred,
- **N** – no data (but the variable is declared),
- **R** – any archive answer did not occurred.

Position of listed characters causes that the character „?”, informing about an error, will not be displayed together with the value. Modifiers of characters determining a style may be combined. To increase the legibility it is recommended to write the modifiers as lower case letter to separate them visually from a style marker. It is illustrated by

EXAMPLE

If the format is declared as below,

:8:3-Lfh,

and if a hole in archive data or a reference to the future occurs, the character **?** will not be output together with a number, and if a string **0?** should be output additionally then an empty place will be output.

After the character denoting a style (**L**, **R**, **C**, **Z**) the O character may be placed. It causes that an expression result will be treated as a number of an operator in the list of the NAME entry of the [OPERATOR] section (counted from 1). Instead of the number the operator name is inserted.

EXAMPLE

The following symbol inserts the name of the first operator.

{S1} ‘1’:10-LO;

Markers

Markers are predefined strings of characters included between the per-cent character ‘**%**’. The markers can be used exclusively inside special type of strings called *lines*. The markers can be delimited by other characters different from ‘**%**’, if these characters will be declared by means of **marker** instruction. It is necessary in case, when the line with markers contains the per-cent character itself. The markers are listed below.

%date%	**%time%**	**%day%**	**%dtime%**
%bdate%	**%edate%**	**%btime%**	**%etime%**
%r*N*%	**%h*N*%**	**%s*N*%**	**%i*N*%**
%page%	**%remark%**	**%operator%**	**%next%**
%print-xxx%			

Where:
N -is a natural number ranging from 0 to n;
xxx - is a command controlling the way of printing (in the graphics mode).

The markers can appear inside the line and they are interpreted by the **format** instruction. Appearing of the marker inside the line causes that in the result report the defined string of characters will replace it. The purpose of individual markers is as follows.

%date% - the day when the report was calculated (format "dd-mm-yyyy");

%time% - hour and minute when the report was calculated (format "hh:mm");.

%day% - the destination day, i.e. the day that the report was calculated for it (for multi-days reports it is the first day of the reporting period);

%bdate% - the date that is related to the beginning moment (format "dd-mm-yyyy");

%edate% - the date that is related to the finish moment (format "dd-mm-yyyy");

%btime% - the time that is related to the beginning moment (format "hh:mm:ss");

%etime% - the time that is related to the finish moment (format "hh:mm:ss");

%dtime% - the time difference between the current finish and beginning moments; whether the difference exceeds one day (24 hours), the string ">1 day" is outputted ";

%r*N*% - the result of evaluation of the N^{th} formula of the **record** instruction (within the format defined by the field and style);

%h*N*% - the N^{th} string of the **header** instruction (within the format defined by the field and style);

%s*N*% **-** the result of evaluation of the N^{th} formula or the N^{th} string of the **symbol** instruction (within the format defined by the field and style);

%i*N*% - the N^{th} question (string) of the **input** instruction is displayed, then the answer (within the format defined by the field and style) is inserted to the report instead of the marker;

%remark% - the operator's note up to 60 characters inserted by means of the dialog window;

%page% - output of the new page on a printer; preserved due to keep compatibility with older versions; at present the **ctrl** instruction should be rather used;

%operator% **-** the operator's name up to 30 characters selected within the dialog window;

%next% - an empty string is always the result of interpretation of the **%next%** marker; the side effect is setting of the new beginning and finishing moment. the **%next%** marker should be used only for definition of reports

that contain the **repeat** instruction; it causes, that newly calculated finish moments becomes the beginning one and the previous finish one is to be restored;

%print-xxx% **-** used to control the way of printing a report in a graphic form. the following commands (xxx) are possible:

PAGE - form feed,
PORTRAIT - toggle of page orientation on vertical one,
LANDSCAPE - sets of page orientation on horizontal one,
FONT(*font_description*) - font changing. Font format description is as follows

font_typeface,hight,width,attributes

The length and width are given in decimal parts of millimeter, accessible attributes are: bold, italic, underline, strikeout. No element of font description is obligatory, particularly the entry FONT() means the use of font according to the lately defined typeface and last sizes with all attributes deleted.

EXAMPLE

Use of markers %print-xxx% for formatting the printout of report content. The sequence:

"%print-landscape%%print-font(50)%tekst zwykły %print-font(bold)% bold %print-font()% zwykły"

has following meaning:

setting the vertical orientation (if not in the report beginning, then it is connected with a page change);
setting a font with default typeface (Courier New or according to the item FONTS or lately; given in the FONT commands) with height of 5 mm and width selected automatically (on the basis of default ratio of arial font);
setting a bold font – the other font parameters as in point 2;
return to the font set in point 1.

Lines

The special type of string, content of them is interpreted by the ASTER language, are called *lines*. They are used for **format** instruction. Just as normal strings, the lines are delimited with quotation marks". They can alternatively be delimited with other characters defined by means of the **delimiter** declaration. The second case is applied, whether the line must contain the quotation marks itself. The lines can include interpreted markers.

EXAMPLE

"Ordinary line"
"Line with the marker %day%"
*The line delimited with the asterix characters can include the quotation mark " *
"The line with the % (per-cent) character contains the marker @day@"

Special Characters

The special characters allowed by the ASTER language are the following: a colon '**:**', a semicolon '**;**' and a comma '**,**'. They can be used as separators (separating characters) for other components.

EXAMPLE

{instruction} symbol {includes}
{the formula} '1+2' : {the colon} 3 {field width} , {the comma}
{the formula} '2+3' : {the colon} 3 {field width} ; {the semicolon}

13.4.2. Instructions

The name Instruction
The export Instruction
The nowarning Instruction
The noerror Instruction
The delimiter Instruction
The marker Instruction
The begin Instruction
The step Instruction
The repeat Instruction
The count Instruction
The header Instruction
The record Instruction
The symbol Instruction
The input Instruction
The format Instruction
The alias Instruction
The ctrl Instruction
The text Instruction
The header, input, symbol, record Instructions

The consecutive instructions of the ASTER language will be described below. This is not the report creation manual, but it is only the list of instructions together with their descriptions.

The name Instruction

Syntax
name *<string>*

The **name** instruction is used for definition of the report name. The name appears on the list of reports. Any string of characters can be used as a name, but no longer than 30 characters are recommended, because the names will be cut off to that length. This is the obligatory instruction. The unique names for individual reports should be used.

EXAMPLE

name "water consumption"

The export Instruction

Syntax
export

The **export** instruction changes the treatment of errors while outputting numbers. When the non-critical error of evaluation occurs then the number without a question mark will be outputted. The error messages that are usually inserted into the areas destined for numbers will not be placed there. This facilities possible further conversion to other formats, DBF for instance.

The nowarning Instruction

Syntax
nowarning

Use of **nowarning** instruction causes that warnings will not be output during reporter operation.

The noerror Instruction

Syntax
noerror

Use of **noerror** instruction causes that error messages will not be output during reporter operation.

The delimiter Instruction

Syntax
delimiter <*char*>

The **delimiter** instruction enables to define an alternative delimiter for lines that appear in the *format* instruction (the quotation mark is the standard delimiter). Definition of the alternative delimiter is necessary, when the quotation mark appears inside the line itself. The **delimiter** instruction, if appeared, should be placed before the format instruction. The newly defined delimiter can be used instead of quotation marks **“”** (but not in the same line).

EXAMPLE

{"line with the ” character"} / is not the correct line
delimiter @ / the @ character will be the delimiter
{@ line with the ” character@}/ is the correct line now

The marker Instruction

Syntax
marker <*char*>

The **marker** instruction enables redefinition of the marker delimiters that appear in the *format* instruction (the per-cent '%' is the standard delimiter). Definition of the alternative delimiter is necessary, when the per-cent '%' character appears inside the line itself. The **marker** instruction, if appears, should be placed before the format instruction.

EXAMPLE

```
{"line with the % character and %remark%"}      / is not the correct line
the @ character                                 / will be the delimiter
marker @                                        / the @ character will
                                                be the delimiter
{"line with the % character and @remark@"}      / line is correct now
```

The begin Instruction

Syntax
begin <*hour*>*[:*<*minute*>*[:*<*second*>*]]*

The **begin** instruction is used for setting the time delay of report starting The starting date is preliminary set by means of a menu. The time defined by the **begin** instruction is added to the beginning time of the report due to establishing of the initial moment of reporting. For instance, the **begin** 2:30 instruction denotes, that beginning of the report will be delayed by 2 hours 30 minutes related to the previously set starting time. The **begin** instruction is optional. Whether missed the zero delay is adopted.

EXAMPLE

```
begin 0:10                      / time delay is 10 minutes
```

The step Instruction

Syntax
step <*hour*>*[:*<*minute*>*[:*<*second*>*]]*

The step instruction is used for definition of the period that the report is referred to. The period defined by the **step** instruction actually refers to one line of the report and can be used multiply for reports that contain many lines (the number of lines is defined, for this case, by the **count** instruction). The finishing moment of the report is determined as the sum of the beginning moment and the period defined by the **step** instruction. As the number of hours is not restricted, the period of several days can be also set. The **step** instruction is not compulsory - whether missed, the period of one hour is adopted.

EXAMPLE

```
step 0:30          / period of 30 minutes
step 72            / period of three days
```

The repeat Instruction

Syntax
repeat

The **repeat** instruction is used when the exact number of lines is unknown in advance. The simplest example of such a report can be the report that contains in consecutive lines the information about exceeding of technological limits. A priori it is unknown, how many times such exceeding will took place during the examined period of time - searching of them should go on until the finishing moment of the report will be reached. Usage of the **repeat** instruction indicates this mode of operation. The **repeat** instruction cannot be used together with the **count** instruction.

The count Instruction

Syntax
count <*number*>
lub
count <*variable*>

The **count** instruction defines number of lines of a report. The instruction is optional, by default the number of lines is set to 1. Every line is referred to the period defined by the **step** instruction.

Instead of the natural number, there can be used the identifier of one of predefined variables that define number of days for the appropriate month or year. Such a variable adopts the value that reflects the number of days in that month (or year), when the report starts. There are four such variables pre-defined in the ASTER language.
days_of_month, days_of_last_month
days_of_year, days_of_last_year

The variable, which denotes number of days of a month, can adopt values 28, 29, 30 and 31.

The variable, which denotes number of days of a year, can adopt values 365 and 366.

The described variables are applied due to create long-period reports (monthly and annual ones), where number of lines is equivalent to the variable number of days within the given period. Due to calculate the report in this way, the beginning moment of the report should be set accordingly (e.g. by means of the list of dates in the report window or the appropriate parameter of the MAKE_REPORT action). The beginning day is to be set as the first day of the covered period.

EXAMPLE

```
count 8                    / the report includes 8 lines
count days_of_month        / the monthly report includes as many lines
                             as there / are days in the current month
```

The header Instruction

Syntax
header *<definition>[,<definition>[...]];*

The **header** instruction enables declaration of a series of strings. The strings, declared within the **header** instruction can be used by the **format** instruction by means of markers that are assigned automatically. The ***%hN%*** markers are assigned to the consecutively defined strings of the **header** instruction list. The **header** instruction is optional. It is usually used for the report header line change without the need of the **format** instruction modification.

EXAMPLE

```
header
"hour"  :11-c,             / the width of a field is 11 characters,

                           / alligned to the centre
"power[W]"  :11-c,         / the width of a field is 11 characters,

                           / alligned to the centre
"temp [°C]"  :11-c;        / the width of a field is 11 characters,

                           / alligned to the centre
```

The record Instruction

Syntax
record *<definition>[,<definition>[...]];*

The **record** instruction enables declaration of a series of formulas that are further used for creation of the reports. The formulas declared within the **record** instruction should be correct expressions of the ASTEL language. The formulas can be used by the **format** instruction by means of markers that are assigned automatically. The ***%rN%*** markers are assigned to the consecutively defined formulas of the **record** instruction list. The **record** instruction is obligatory - it must be used.

EXAMPLE

```
record
'hour (btime)'  :11-c,     / the width of a field is 11 digits, aligned to the
                             centre
'avg(power)'    :11:3-c    / 11 digits, three decimal fractional ones
'avg(temp)'     :11:3-c    / 11 digits, three decimal fractional ones
```

Reports not using the **record** instruction may be compiled without any obstacles. The user will be warned with an appropriate message about the lack of the **record** instruction.

Report pattern syntactics (file *.r) gives a possibility of easy defining of reports, which are not based on cyclic calculations with a set time step without using „tricks", like „empty record instructions", what might lead, in case of monthly reports, to unwanted side effects (outputting 30 empty lines – that is an empty sheet on the end of printout).

The symbol Instruction

Syntax
symbol *<definition>[,<definition>[...]];*

The **symbol** instruction enables declaration of a series of formulas or strings that are further used in the **format** instruction. The ***%sN%*** markers are assigned to the consecutive declarations that appear on the **symbol** instruction list. When a given marker appears in any line of the **format** instruction, the appropriate formula will be calculated and either the result of calculation or the declared text will be placed in the report. The beginning time of the report is the default initial moment for all the formulas from the **symbol** instruction, as well as the finishing time of the report constitutes the terminating time of formulas. The **symbol** instruction is optional, although it must appear whether the ***%sN%*** markers have occurred in any line of the **format** instruction.

EXAMPLE

symbol 'avg(x)+avg(y)':10:3;; /'avg(x)+avg(y)' is assigned to the **%s0%** marker, / whenever the /%s0% marker appears, the sum of the average / value of the x variable and the average value of the y / variable will be calculated and the result will be inserted into / the 10 digits wide field with 3 decimal fractional digits

The input Instruction

Syntax
input *<definition>[,<definition>[...]];*

The **input** instruction enables the operator to insert into the contents of the report those text and numerical constants that cannot be read from the archive or obtained from controllers, but may be important from the point of view of document flow within a factory. The strings of characters that stand for the order numbers, material symbols, tool identifiers etc. can serve as examples of such information. The strings are entered as result of interpretation of the ***%iN%*** markers that have been assigned to every definition occurring in the **input** instruction. The question for the data that is to be entered will be displayed before. The entered string is formatted following the field and the style.

EXAMPLE

```
input
"enter the order number?":10,/the answer will be assigned to the %i0%
marker
"enter the material name?":10-c;      /the answer will be assigned to the
%i1% marker

   /whether the marker %i0% occur, the question "enter the order
   /number?" will be displayed. The answer for this question will be
   /inserted into the report for the place of the marker into the
   /10-character wide field (aligned to the left). Occurring of the
   /%i1% marker cause displaying the question: "enter the material
   /name?":, The answer for that question will be inserted into the
   /report for the place of the marker into the 10-character wide  /field (in
the center)
```

The format Instruction

Syntax
format *<line>[,<line>[...]];*

The **format** instruction is used for making reports. Any report can be created by means of it.

```
The algorithm of this instruction is as follows:
until there are lines available repeat
  read a line
    begin
      until the line is not empty repeat
        begin
          extract a component of the line
            if the component is a marker
                        then interpret the marker
                        else {the component was a character} write
to the
                         report
        end
    end
```

Interpretation of the marker consists in insertion the appropriate string of characters into the line, inside the location when the marker was. The line, where the ***%rN%*** markers have been specified is interpreted as many times, as it is the number of report lines defined by the **count** instruction. For the reports, where the **repeat** instruction has been used, the line containing that instruction is repeated until the finishing moment of the report will be reached.

There is a possibility to use a more compact writing of markers generated by header, input, symbol and record:

Instead of: %r0%%r1%%r2%%r3% the sequence: %r1..3% may be written.

EXAMPLE

```
{the simply report without a border}
symbol {%s0} "Title";
header
{%h0}    "hour"            :11-c,
{%h1}    "power [W]"       :11-c,
{%h2}    "temp [°C]"       :11-c;
record
{%r0}    'hour(btime)'     :11-c,
{%r1}    'avg(power)'      :11:3-c,
{%r2}    'avg(temp)'       :11:3-c;
format
"%s0%",                    /title


"%r0%%r1%%r2%";            /formulas
```

The alias Instruction

Syntax
alias *<string>*
or
@ *<string>*

The **alias** allows the strings defined in the initialization file to be inserted into the report. Thus, identical report definitions can be used for report creation, which vary in certain texts, for example in their titles. The instruction argument is the name of the text equivalent, which should be defined in the [TEXT_PARAMETERS] section of the initialization file. The **alias** can be placed in the **symbol** instruction instead of the string. An **@** character can be used instead of the **alias** password.

EXAMPLE

if the following section is inserted in the initialization file
[TEXT_PARAMETERS]
NAME=EXHAUST GASES FAN DRIVE No 3
then, in the report definition you can use the instruction
symbol {%s0} alias "NAME";

The ctrl Instruction

Syntax
ctrl *<string>*
or
^ *<string>*

The **ctrl** allows the printer control codes defined in the initialization file to be inserted into the report. Thus, the printer performance may be fully utilized. Replacing the printer or duplicating the application on another computer system equipped with a printer of different type requires only the initialization file contents to be edited. *Italic* and **bold** types as well as color attributes can be used for fonts. Functioning of the **ctrl** instruction is similar to that of the **alias** instruction. Nevertheless, two differences exist.

Firstly, the text equivalents (i.e. the printer control codes) are defined in the initialization file as hexadecimal character codes separated with space characters. Execution of the **ctrl** instruction causes interpretation of such string where the hexadecimal codes are replaced with appropriate characters. Such sequence of the control codes is to be inserted in the report before printing with the printer using the PRINT function from the REPORT menu of the report window. The control codes instead are not displayed and their effects are not displayed too. **ctrl** can be placed in the **symbol** instruction instead of a string. A **^** character can be used instead of the **ctrl** password.

EXAMPLE

if the following section is inserted in the initialization file (the exemplary codes for HP printers)
[TEXT_PARAMETERS]
BOLD= 1B 28 73 33 42
NORMAL= 1B 28 73 30 42
then, in the report definition you can use the instruction
symbol
{%s0} ctrl "BOLD",
{%s1} ctrl "NORMAL";
format
/the report title will be bold typed
"%s0% report title %s1%",
. . .
Interpretation of the **%s0%** marker will cause that the control code is sent to the printer, which will activate the boldface. This code is written to the initialization file as a text equivalent named **BOLD**. Writing in boldface terminates after the **%s1%** code is read (normal font).

If the printer cannot, for example, use the boldface, it is not required to eliminate "**BOLD**" from the report definition with its associated marker in the **format** instruction. It is enough to either eliminate the line, which describes the BOLD text equivalent from the initialization file, or to not associate any control codes.

The text Instruction

Syntax
text <*string*>
or
<*string*>

The **text** allows the strings stored in the ASMEN variables to be inserted into the report. Thus, identical report definitions can be used to create reports, which vary in certain texts. Such text might be for example, alphanumeric fields stored in the database. The instruction argument is the variable name. **text** can be placed in the **symbol** instruction instead of a string. A **#** character can be used instead of the **text** password.

☑ ***The header, input, symbol, record Instructions***

Instructions, which don't introduce any variable, don't output any empty line too.

13.4.3. The ASTER Language Grammar

<character> ::=
<characters> ::=
|<character>|<characters><character>
<string> ::= **"**<characters>**"**
<formula> ::= **'**<expression>**'**
<style> ::= |**-L**|**-R**|**-C**|**-Z**
<marker> ::= **%date%**|**%time%**|**%day%**|

%dtime%|**%bdate%**|**%edate%**|**%btime%**|**%etime%**|

%r<integer>**%**|**%h**<integer>**%**|**%s**<integer>**%**|**%i**<integer>**%**|

%page%|%remark%|**%operator%**|**%next%|%print-xxx%**
<element> ::= <character>|<marker>
<elements> ::= |<element>|<elements><element>
< line > ::= **"**<elements>**"**
< comment > ::=
{<characters**>}**|**/**<characters><carriage_return_character>
<instruction> ::= <name>|<delimiter>|<marker>|

<begin>|<step>|**repeat**|<count>

<symbol>|<format>|<header>|<record>|
<input>|
export|<alias>|<text>|<ctrl>|**nowarning**|**noerror**
<name> ::= **name** <string>
<delimiter> ::= **delimiter** <character>
<marker> ::= **marker** <character>
<count> ::= **count**
<integer>|**count**<**days_of_month**>|

count<**days_of_last_month**>|**count**<**days_of_year**>|
count<**days_of_last_year**>
<begin> ::= **begin** <hours>|
begin <hours>**:**<minutes>|
begin
<hours>**:**<minutes>**:**<seconds>
<step> ::= **step** <integer>|
step <integer>**:**<minutes>|

```
                                        step
                                    <integer>:<minutes>:<seconds>
<field>         ::=         |:<integer>|:<integer>:<integer>
<hdlist>        ::=
                    <string><field><style>|<hdlist>,<string>
                    <field><style>
<header>        ::=         header <hdlist>;
<rdlist>        ::=
                    <formula><field><style>|<rdlist>,<formula
                    ><field><style>
<record>        ::=         record <rdlist>;
<alias>         ::=         alias<string>|@<string>
<text>          ::=         text<string>|#<string>
<ctrl>          ::=         ctrl<string>|^<string>
<sblist>        ::=
                    <string><field><style>|<formula><field>
                    <style>|

                    <sblist>,<string><field><style>|<sblist>,
                    <formula><field><style>|

                    <sblist>,<alias><field><style>|<sblist>,<
                    ctrl><field><style>
<symbol>        ::=         symbol <sblist>;
<input>         ::=         input <hdlist>;
<frlist>        ::=         <line>|<frlist>,<line>
<format>        ::=         format <frlist>;
<report>        ::=         <instruction>|<report><instruction>
```

The ASTER language grammar has been written with the use of the BNF notation. It is not so precise notation, as the ASTER language rules cannot be written in the formal mode under a notation of this type because they are non-context rules (a notation for a double level grammar should be used). Inaccuracies occur for the definitions of the string and the line. It is caused by the possibility of redefinition of delimiters for both the string and the line. The <character> rule denotes all the characters that can be loaded down from the keyboard. The characters that create a string cannot contain the ” character unless the string delimiter has been redefined.

13.4.4. Diagnostics

Syntax Errors
Semantic Errors
Other Errors
Warnings

The report definition is analyzed before the report itself is generated. During the analysis, some errors may be detected, which make generation impossible. If a syntax error is detected, the analysis is interrupted and an error message is generated. Upon completion of the syntax analysis (and partially during the analysis as well), the semantic analysis of the report definition is accomplished. The resulting possible semantic errors may also disable the report generation. Also, situations can be detected which potentially might generate erroneous reports. They only produce warnings before generating the report.

If syntax errors are detected, the information is produced on the error type and location (LINE). Such information will be written in a special window. The diagnostic information produced by the ASTER language interpreter has the following form:

Error in line <line number> word <word number>
LINE: <erroneous line with the erroneous ASTER language element indicated with the question mark '?'>
ERROR: <error description>

If a syntax error is detected inside the formula, the line is produced twice. First, the ASTER language checker displays the erroneous line with the invalid formula indicated with the question mark '?'. Then, the ASTER language checker displays the erroneous expression with the invalid element preceded by the question mark '?'. The information has the following form:

Error in line<line number> word <word number>
LINE: <erroneous line with the erroneous ASTER language formula indicated with the question mark '?'>
ERROR: <error description>
FORMULA: <the ASTEL language expression with the erroneous element preceded by the question mark '?'>

If no syntax error but only semantic errors are detected, then the following information appears:

Error in line <the last line number> word <the last word number>
ERROR: <error description>

If no errors but only warnings are detected, then the following information appears:

Line count <the last line number>
WARNING: <warning message>

If no errors and no warnings are detected, then the following information appears:

Line count <the last line number>
no errors

Syntax Errors

The syntax errors are signaled by one of the following messages. Additionally, there is produced the erroneous line with the invalid element indicated (each line may contain multiple elements of the same type, for example the formulas, therefore the message itself does not indicate the error cause precisely). There are two groups of errors: the fact that the checker expects an another type of element in a given place (the message of type **expected...)** or the element itself is erroneous although its type is correct (other messages).

Below there are listed all messages related to the syntax errors.

keyword expected
illegal keyword
formula expected
formula or string expected
invalid or too long formula
string expected
string too long
number expected

invalid integer
semicolon or comma expected
semicolon, comma or minus expected
semicolon, comma, colon or minus expected
style expected
sign expected

Semantic Errors

The semantic errors are detected mainly after the syntax analysis is completed. They mean, that the report definition is incomplete, discrepant or redundant (some instructions are redundant). Potential erroneous situations (with comments included) are listed below:

no report name The report name is omitted in the definition.

statement redefinition An instruction is repeated (e.g. count 5 count 10).

bad range of number The number is correct, but its range is improper (e.g. hours=30).

invalid value in COUNT The count instruction has its argument 0 (it means 0 lines thus a blank report).

RECORD undefined The record instruction is omitted in the definition.

FORMAT undefined The format instruction is omitted in the definition.

COUNT and REPEAT defined The count and the repeat instructions eliminate each other.

invalid symbol in FORMAT An unknown/undefined marker is detected in the format instruction.

RH fields mixed in FORMAT line The markers from the record and header instructions are inserted in the same line of the format instruction.

more than one line with RH fields The markers from the record and header instructions are placed more than in one line only.

Other Errors

no memory The report definition is too long, the memory is insufficient for interpretation. Separate the definition in two simpler ones if practicable.

Warnings

The warnings do not disable the report generation. They indicate instead, that the report's appearance might mismatch the user's expectations. Warning messages (with comments included) are listed below:

HEADER and RECORD conflict It occurs if the list of the strings declared in the header instruction is not number-compatible with the list of the formulas declared in the record instruction. Also it may occur if the list of the markers related to the record instruction and existing in the format

instruction is not compatible with the markers related to the header instruction.

time shift greater than 1 day It occurs, if the begin instruction argument is greater/equal 1 day (24 hours). The side effect could be, for example, that the 1st January report would contain data from the 2nd January.

13.5. Script Reports (VBScript, JavaScript)

A standard language for creating reports in **asix** system is ASTER the interpreter of which is built-in into the AS program. Reports can also be written in script languages: VBScript and Jscript. This allows creating reports the calculation method and form of which cannot be achieved in the ASTER language, e.g. various external data sources may be used.

The AS program enables integrated handling of both types of reports.

13.5.1. Declaring Script Reports

Unlike the reports prepared in the ASTER language, every script report that is used must be declared in the application configuration file overtly.

Declaration of script report is carried out with use of Architect program.

See more detailed information in:
Architect user's manual, chapter *3.13.1. Declaration of script reports files.*

13.5.2. Raporter Window

Handling of reports written with use of scripts does not differ essentially from handling of reports created in the ASTER language. The list of reports contains the names of both types. They are not distinguished in any way. The only differences in handling are:

- for script reports the command *Verify* from the Raporter window's menu is not executed;
- the command *Directory* is applicable to reports in the ASTER language only. Regardless of which directory is selected, the collection of displayed script reports is always the same and results from declarations saved in the ini file only.

13.5.3. Operator Actions

In the set of operator actions, you will find three actions designed for handling reports. The se are: MAKE_REPORT, PRINT_REPORT and SHOW_REPORT.

If an action is to concern a report prepared in the ASTER language, then in the first parameter you should pass the file name (with *.r* extension) containing the report definition.

If an operator action is to concern a script report, then in the first parameter you should pass the script report name under which it was declared in the ini file. Script

reports used in operator actions must be also declared in the ini file. If the report name includes spaces and appears in the contents of operator action, the name should be put in quotation marks ".

13.5.4. Rules of Creating Script Reports

Scripts used for report creation are provided with a few characteristic features that differentiate them from other scripts used in **asix** system. Report scripts create the report contents in the form of text-type files. Upon calling, a set of parameters defining the report generation method is transferred to the script. The first parameter is always the name of text-type file which the generated report is to be saved into. Below there is a script that writes the values of all transferred parameters into the report file:

```
dim fso, file
set fso = CreateObject("Scripting.FileSystemObject")
set file = fso.CreateTextFile(asix.script.arguments(1), True)

for each p in asix.script.arguments
        file.WriteLine p.name & " = " & p.value
next
file.Close
```

13.5.5. Meaning of Script Parameters

The following parameters are transferred to scripts that generate reports:

Table 13-4. The list of Parameters Transferred to Scripts that Generate Reports.

Number	Name	Type	Meaning
1	OutputFile	Text	Name of file the generated report is to be placed in.
2	ReportTime	Date	Date and hour for which a report is to be generated. This parameter value results from selection made by operator in the Raporter window or from operator action parameters.
3	ForPrint	Number	Defines the purpose of report generation. Value 1 means generation of report to be printed, value 0 means generation of report to be displayed on the screen.
4	Parameters	Text	Parameters of report generation – copied from *report_parameters* parameter given in the SCRIPT item, which declares the script report.
5	Note	Text	Contents of the note entered in Raporter window or in parameters of operator action.
6	Operator	Text	Name of the operator selected in Raporter window or provided in parameters of operator action.

13.5.6. Printout Control

If the report is generated for printing, then you may insert into the contents of created report the printout formatting instructions. They are compatible with formatting by means of PRINT-xxx markers of reports prepared in the ASTER language.

EXAMPLE

The following example allows a fragment of report to be printed in bold font.

```
if asix.script.arguments(3) then
file.Writeline CHR(27) & "FONT(bold)" & "Temperatures" & CHR(27) & FONT()"
else
        file.Writeline "Temperatures"
end if
```

The printout control instructions should be used only if the *ForPrint* parameter is equal to 1. Instruction always starts with Escape character followed by instruction contents. The following variants are possible:

- PAGE
  Feed paper.
- PORTRAIT
  Set page orientation to vertical.
- LANDSCAPE
  Set page orientation to horizontal.
- FONT(font_description)
  Font change. Description of font format is as follows:

  typeface_name,height,width,attributes

Height and *width* are given with accuracy to tenth part of millimetre, the available *attributes* are: bold, italic, underline, strikeout. No element of font description is obligatory; in particular, the FONT() notation means that font will be used according to last defined typeface and sizes, with all attributes removed.

13.5.7. Codepage of Object Files

For displaying and printing reports the **asix** system uses fonts with selected codepage of OEM type. This font depends on operating system. For system configured for operation in the Polish language, it is codepage no 852. The OEM page is used by operating system in the character console windows. For generation of texts containing national characters, you should remember to introduce them in the standard of OEM page being used.

The characteristic feature of OEM fonts is presence of characters used for creating frames.

14. Multi-Monitor Systems

Hardware Configuration
Opening Actions
Interpreting Keyboard Shortcuts
Operation with Standard License

The **asix** system is provided with built-in functions that enable designing the applications designed for operation on multi-monitor computers.

These functions are available after purchasing the **asix**'s up-grade license for multi-monitor operation.

Hardware Configuration

The first step that is necessary for starting-up a multi-monitor system is correct configuration of its hardware. For Windows ME and Windows 2000 handling of multi-monitor operation is built-in to the operating system. All what is needed is using the graphic boards compatible to the operating system. In the case of Windows NT, multi-monitor operation is not supported by the operating system. It is necessary to use the graphic boards for which the manufacturer supplies drivers enabling multi-monitor operation under control of Windows NT.

Under control of Windowsm ME and Windows 2000 systems, **asix** enables operation for any configuration of displays. In the case of Windows NT only configuration of 2-4 displays in horizontal arrangement is possible.

Opening Actions

In multi-monitor mode displaying the synoptic masks, tables and trends on selected display monitor is possible. To specify this the *x, y and monitor* parameters in operator's actions OPEN_MASK, TABLE and ASTREND are used.

The x and y parameters permit on direct indication of target display where the windows is to be opened. However, it is recommended to use these parameters to specify the position on the primary display only, but for display selection to use the *monitor* parameter**.** *monitor* parameter may take one of the following values:

- monitor number - the opening coordinates are calculated in order to display window on selected monitor; he displacement according to left upper corner of main monitor is taken into account;

- 0 - the opening coordinates are calculated in order to display window on the monitor selected by mouse pointer; he displacement according to left upper corner of main monitor is taken into account. This mode enables on dynamic adapting of application to operator's actions;

- -1 - coordinates on monitor screen are not recalculated; window will be opened at position given at action parameters or in definition of the synoptic mask.

Additional feature in multi-monitor operation is change in operation of opening modes connected to exchange of windows. Only those windows are closed that are displayed (even partially) on the screen on which the new window is to be displayed.

In all cases **asix** prevents for attempts to open the window in area not covered by any monitor. In such a case the coordinates of window are modified in order to open it on main monitor. Application designed for multi-monitor, operating in single-monitor system will display all windows on the main monitor.

Interpreting the Keyboard Shortcuts

During the running the application pressing the key on the keyboard triggers the mechanism of interpretation of keyboard shortcuts. All open synoptic masks are searched (in objects definitions or keyboard shortcuts connected to masks) until connection of key to operator's action is found. In normal mode all opened masks are examined. Using the parameter ***Keys by screen***:

Architect > *Fields and Computers* > *Miscellaneous* module > *Misc* tab

you may specify that only masks are searched, that are displayed (even partially) on screen indicated by mouse pointer. This feature permits on different interpretation of key depending on screen used in this moment by the operator.

Operation with Standard License

In the case where you have the standard license of **asix** only, using the multi-monitor station involves necessity of defining the coordinates indicating the proper display in the parameters for masks or in opening actions. Only protection for attempt to open a synoptic mask in area not covered by any monitor is active.

15. Authorization Control System

The **asix** package includes mechanisms that allow to provide applications with a user authorization system.

Operations controlling the process as well as actions controlling the **asix** application operation are objects of protection.

Thanks to a hierarchical password system it is possible to declare for all visualization objects, that can execute controlling operations, the level of its protection. Each of levels identified by the number from the range 1-4 (with 5th additional administrator level when integration of the password system with logon system is applied) has its access password.

Operator actions controlling the application operation (actions and some commands accessible form menu) are protected with individal passwords. Setting parameters for such a protection consists in defining the proper position in an application XML file as well as the proper selection of actions being used.

Detailed informations on parametrization of the password system are described in the chapter *9.5. Passwords.*

A more advanced method of the user authorization system for **asix** applications is a logon system. This solution allows to assign passwords, that authorize for realization of specified actions, to specific users. Data concerning all users and their authorities are stored in a database placed in a text file with the .ini extension. The operation of user logging of 1-5 level corresponds to entering the password permanently. Moreover, the logon system makes it possible to trace all operations of logging in, logging out and threefold incorrect input of a password. These events are written into alarm log files. The logon system also assures the access to the data of a currently logged user.

The **asix** software assures full compatibility of the password and logon systems (when the logon system is used).

Detailed information on parametrization of the logon system are described in the chapter *9.6.Logon System.*

16. Operator Actions

Operator actions are a basic way of the operator control of the application operation. The actions can be accessed (by a system designer) with the use of objects of the class such as BUTTON, Key shortcuts or arbitrary defined menus.

Actions are described by means of text strings. As a rule the action descriptions are entered in relevant edit fields. These fields format automatically any text if it is correctly entered. The action description consists of an action name and its parameters. Individual elements of the description are separated with spaces or commas. The number and type of parameters depend on the action type. In some cases it is possible to omit certain parameters, which means that default values are taken up. There are five types of parameters: text, symbol (a text chosen from an allowed set), integer, floating-point number, hexadecimal number.

The character case is not meaningful for the action names and symbols (characters are converted into capitals). The letters type may be of importance only for some text parameters. In case of both action names and symbolic parameters it is sufficient to enter some initial letters identifying explicitly an action or symbol.

Sometimes the action descriptions are declared in text files without using the formatting fields (this concerns the case of action sets). However, principles of the action descriptions are not subject to change. There is no need to use fully formatted descriptions. Each description form that would be acceptable by the formatting field is allowed and will be carried out correctly.

REMARK *The text parameters should be put in quotation marks (“”), that allows to insert a space character into the parameter.*

It is possible to use Action Editor called by clicking right-button of a mouse in edition field where the action is entered.

16.1. Operator Actions Reference List

In this chapter, the following action types are described:

ACTION_SET	- carrying out an arbitrary action set
ASBASE	- enables AsBase program operation to be remotely controlled
ASTREND	- control of ASTREND program operation
ASAUDIT	- control of displaying AsAudit windows
ACKNOWLEDGE_ALARMS	- confirm of alarms on list of active alarms
ASK	- open window with a question to an operator
CHANGE_BITS	- change the value of chosen bits of a process variable
CLEAR_SOUND	- clear sound alarm

CLOSE_MASK - close diagram
CONNECT ALARMS - connect to the specified source of historical alarms
CURSOR - adjusting of a cursor position
DIALOGS - opening the dialogs window
DISPLAY_FILE - display file contents
DRAW_TREND - display trend window
EXCLUDE - change alarm excluding status
EXIT - the AS program work completion
FILE_PRINT - file printing
GET_PASSWORD - asking an operator for the password
HIDE_ALL - minimize application to the icon
HISTORY_READ - historical data complement from the controllers
LANGUAGE - switching the application language
LOGON - logging system
LOGOFF - logging off a current user
MACRO - macro replay
MAKE_REPORT - report making
MENU - opening the defined menu
NEW_TIME - open time updating box
NOTHING - empty action
OPEN_MASK - display diagram
PANEL - Control Panel selection
PASSWORDS - open of the password manager window
PAUSE - postpone a program action
PERFORM_INPUT - an order to objects on synoptic diagrams to send control value
PLAY - WAV type file replay
PRINT_REPORT - report calculation and printing
REPORTS - open of the reports system window
RUN - program start up
SCREEN_PRINT - printing all screen or active window
SCRIPT - request of script execution
SEND_VALUE - send a number value to the controller
SERVER_FILTER - locking the access to network stations
SET_BITS - send a bit value to the controller
SET_CHART - set the time of CHART object
SHOW_REPORT - report calculation and display
SIGNALS - selection of alarms the appearance of which results in sound signalling
TABLE - open variable table window
TRANSFER - transfer value of process variable
TRENDS - display trend window
VARIABLE_DESCRIPTION - display information about indicated variable

16.2. Description of Actions

ACTION_SET
ASBASE
ASTREND
ASAUDIT
ACKNOWLEDGE_ALARMS
ASK
CHANGE_BITS
CLEAR_SOUND
CLOSE_MASK
CONNECT_ALARMS
CURSOR
DIALOGS
DISPLAY_FILE
DRAW_TREND
EXCLUDE
EXIT
FILE_PRINT
GET_PASSWORD
HIDE_ALL
HISTORY_READ
LANGUAGE
LOGON
LOGOFF
MACRO
MAKE_REPORT
MENU
NEW_TIME
NOTHING
OPEN_MASK
PANEL
PASSWORDS
PAUSE
PERFORM_INPUT
PLAY
PRINT_REPORT
REPORTS
RUN
SCREEN_PRINT
SCRIPT
SEND_VALUE
SERVER_FILTER
SET_BITS
SET_CHART
SHOW_REPORT
SIGNALS
SYNCHRONIZE_ARCHIVE
TABLE
TRANSFER
TRENDS
VARIABLE_DESCRIPTION

ACTION SET action_name[, text]

Abbreviation - **AC**

Operation type - command to carry out an action set that is a sequence of any arbitrary set of operator actions. This action allows attaching many operations to one operator action.

Parameter *action_name*

Meaning - name of an action set to localize component actions. The action set is performed in two steps. In the first step the items are searched for name consistent with the action name declared in ***Text parameters*** parameter (Architect > *Fields and Computers* > *Masks* module > *Text parameters* tab). The actions given in all found items are run. If such entries have not been found, an attempt will be made to read a file with a given action name and extension *ACT*. If such a file exists, then all the actions described in it will be carried out. The action file is a common text file, in which each line describes one action. The searching is being carried out in all the folders of visualization diagrams.

Type - text

Parameter *text*

Meaning - any text, which is entered to the content of component actions in the place of „%s" string. It allows creating actions, whose content may be changed.

Type - text

ASBASE operation_type[operation_parameters]

Abbreviation - **ASB**

Operation type - enables AsBase program operation to be remotely controlled.

Symbolic parameter may have the following values:

STart

Meaning - the action forces AsBase program start with all parameters declared in *operation_parameters* (usually xml file of application). If AsBase was already run, Start operation is equivalent to Show operation without parameters. Additionally, during AS program shut-down, AsBase also ends its operation.

SHow

Meaning - the operation forces AsBase window activation (displayes the window on the top of other windows). Description of the information is to be displayed in the window may be optionally set in *application_parameters*. The structure of the parameters is as follows:

*Object_type**object_identifier*[*set_identifier*]

where:
object_type may be equal to:

RCP - request of recipe table content displaying or viewing of variable set conencted with recipe group;
RLD - request of displaying the table of recipe loading history;
ARC - request of displaying the table of registration set archive or viewing of variable set connected with registration set;

object_identifier – identifier of registration set or recipe group;
set_identifier - optional, identifier of variable set;

The *Show* command allows to choose the view. The structure of the parameters is as follows:

*object_type**object_identifier**view_identifier*

view_identifier – identifies the set of parameters that determine the way of displaying the content of a given table.

LOad

Meaning - the operation enables recipes to be loaded. The structure of operation parameters is as follows:

*group_identifier**set_identifier*[*recipe_name*]

where:
group_identifier and *set_identifier* determines: group of recipes the operation is refers to and set of variables where recipe is to be loaded. If *recipe_name* is declared, the action loades indicated recipe to variable set. If *recipe_name* is not declared, the window with all the recipes defined in the group, from which the operator will be able to choose the one to be loaded, will be displayed.

NAvigate

Meaning - the operation enables to browse tables of an AsBase module database on **asix** application masks. The structure of the parameters is as follows:

*connection_identifier**command*[*filter_definition*]

where:
connection_identifier – the connection identifier defined during the ASBASE object parameterization;
command – the name of the command that has to be performed:
FIRST – displays the first record of a table,
NEXT - displays the next record of a table,
PREVIUS - displays the previous record of a table,
LAST - displays the last record of a table,
FILTER – defines a new filter and displays the last record of a table compatible with the filter,
COUNT – assignes a set of variables of the number that is a quantity of table records compatible with a current filter to a particular variable;
filter_definition – definition of a new filter; this parametr is passed only for the FILTER command; the filter definition should be designed according to the rules described in ASBASE Object.

PRint
Meaning - the operation enables to print the content of tables of an AsBase module database. The structure of the parameters is as follows:

*object_type**object_identifier*[*view_identifier*]

where:
object_type – can be:
RCP – table of recipe;
RLD – table of recipe loading history;
ARC – table of registration set archive;
object_identifier – identifier of registration set or recipe group;
view_identifier – optional identifier of one of the views connected with a given table.

ADd
Meaning - the operation enables to add a new record to the content of an AsBase database table. The structure of the parameters is as follows:

connection_identifier

where:
connection_identifier – the identifier defined when configuring the ASBASE object (see: *18.4.1. ASBASE Object).*

DElete
Meaning - the operation enables to remove a record from the content of an AsBase database table. The structure of the parameters is as follows:

connection_identifier

where:

connection_identifier – the identifier defined when configuring the ASBASE object (see: *18.4.1. ASBASE Object)*.

UPdate

Meaning - the operation enables to update the current record in an AsBase database table. The structure of the parameters is as follows:

connection_identifier

where:
connection_identifier – the identifier defined when configuring the ASBASE object (see: *18.4.1. ASBASE Object)*.

☑ *ASTREND trend[, x, y, monitor, directory, Variable Database]*

Abbreviation - **AST**

Operation type - opening a window of ASTREND program with simultaneous trend display according to the definition written in *.trn a file.

Parameter *trend*

Meaning - file name with default extension **trn**, which contains previously prepared definition of the trend displayed by the ASTREND program. Entry of the character „?" causes opening a window where user can select a trend to display.

> The *trend* filename may be appended by a list of names of variables that are to be displayed. Names on the list must be separated by the # character.

Type - text

Parameters *x,y*

Meaning - x,y coordinates of the upper left upper of ASTREND program window. A value of –1 means taking a proper coordinates from the trend definition. The item is used in multi-monitor systems.

Type - decimal number

Default value - –1

Parameter *monitor*

Meaning - indication of monitor where an ASTREND program window should open (concerns multi-monitor system). A value of 0 means a monitor where currently the mouse cursor is; –1 means a monitor resulting from assumed x, y parameters. The item applies to multi-monitor systems.

Type - decimal number

Default value - –1

Parameter *directory*

Meaning - directory, from which a trend definition is to be read (file *.**trn**).

Type - text

Parameter *Variable Database*

Meaning - Variable Database name (set of variables, file *.**zez**), which is to be loaded.

Type - text

ASAUDIT window_type, window_parameters

Abbreviation - ASW
Operation type - control of displaying AsAudit windows called from asix application
Parameter *window_type*
Meaning - name of the window to be displayed:
Console
Config
Browser
Type - text
Parameter *window_parameters*
Meaning - the parameter is not used at present

FILE_PRINT file_name

Abbreviation - **FIL**
Operation type - order to print a file
Parameter *file_name*
Meaning - name of the file to print
Type - text

SERVER_FILTER server_name

Abbreviation - **SER**
Operation type - information for ASLINK module defining stations it can be linked to.
Parameter *server_name*
Meaning - a network name of the computer to link with. To designate a group of computers, signs * and ? (wildcard) can be used.
Type - text

REMARKS *An action only for specific applications if you wish to control a source of data origin.*

PLAY [file_name]

Abbreviation - **PL**
Operation type - playing the WAV type file.
Parameter *file_name*
Meaning - the name of WAV type file whose contents is to be played.
Type - text
Default value - an empty parameter causes an earlier started play to be interrupted.

REMARKS *A sound board must be installed in a computer.*

GET_PASSWORD password

Abbreviation - **GET**
Operation type - action provided to be used as a component element of action sets. If an operator does not specify the correct password, completion of an action set will be immediately interrupted.
Parameter *password*
Meaning - password
Type - text

EXIT [password]

Abbreviation - **EX**
Operation type - termination of the AS program operation. This action is equivalent to calling MASKS.EXIT command and closing the Control Panel.
Parameter *password*
Meaning - the password content, that shall be known to complete (close) an application
Type - text
Default value - no password entered denotes applying the password given in EXIT_LOCK entry of the initial file or if this item has not been used, an exit without password checking.

REMARKS *This action allows to end a program work, even if the parameter **Exit lock** parameter has been used in the configuration file. Password content may be different from passwords declared for the level 1-4 and cannot be changed from the application level.*

HISTORY_READ

Abbreviation - **HIST**
Option type - the action forces a complement of historical data from the drivers apart from the cycle declared in ***Time of first merging*** and ***Merging frequency*** parameters in the configuration file (Architect > *Fields and Computers* > *Historical data* module > *Historical data* tab); the operation of data complement is performed by the ASPAD module.

LANGUAGE [language_code]

Abbreviation - **LAN**
Option type - performing the action (with declared code) causes language switching. Lack of the parameter causes opening the window for language selection.
Parameter *language_code*
Meaning - 2-character language code; it is recommended to keep capability with ISO639 standard; language code is used to select active language and choose the proper text version used in application; instead of the parameter it is possible to insert a code that has been declared in application configuration file with use of Architect

(Architect > *Fields and Computers* > *Start parameters* module> *Languages* tab);
e.g.: pl – Polish, en - English.
Type - text
Default value - lack

☑ ***LOGON***

Abbreviation - **LOG**
Option type - the action opening the window used to **asix** application user logon (when using the logging system).

☑ ***LOGOFF***

Abbreviation - **LOGOF**
Option type - the action causes the immediate operation of logging off current user or switching over to default user.

☑ ***CURSOR x, y***

Abbreviation - **CUR**
Operation type - adjusting a mouse cursor in a preset position
Parameters *x, y*
Meaning - new position of the mouse
Type - integer

REMARKS *An action provided to create presentation.*

☑ ***MAKE_REPORT file moment [operator] [note]***

Abbreviation - **MAK**
Operation type - making a specified report. The complete report will be saved on a disk.
Parameter *file*
Meaning - file name with a report definition (full path)
Type - text
Parameter *moment*
Meaning - a moment of report starting in ASTEL language convention
Type - text
Parameter *operator*
Meaning - operator's name (if it appears in the report definition)
Type - text
Default value - no name
Parameter *note*
Meaning - content of a note (if it appears in the report definition)
Type - text
Default value - no note

☑ ***MACRO file_name***

Abbreviation - **MAC**
Operation type - execute macro (demo sequence) saved in a file. While being replayed all time dependence are conserved.
Parameter *file_name*

Meaning - name of a file, which contains a macro. File should be created by recording the demo sequence (OPTIONS.DEMO.RECORD command) and then by renaming the file replayw.dem
Type - text

REMARKS *The action provided to create presentation.*

☑ *MENU file_name [x][y]*

Abbreviation - **ME**
Operation type - this action displays pop-up menu, the form of which is freely defined with use of Architect program.
Parameter *file_name*
Meaning - name of created menu.
Type - text
Parameters *x y*
Meaning - a place where the menu should be created
Type - integer
Default value - depends on the context of an action use. If called from BUTTON object, so the object position is accepted. Otherwise, coordinates (100,100) are assumed.

See: Architect user's manual, chapter *3.5.3. Definition of menu located on application mask*.

☑ *NOTHING*

Abbreviation - **NO**
Operation type - empty action

REMARKS *This action is provided for locking of keyboard actions defined on other diagrams. If key set on the active diagram contains NOTHING action then any keyboard actions (for chosen key shortcut) from other diagram are not being carried out.*

☑ *DIALOGS [x] [y]*

Abbreviation - **DIA**
Operation type - opening the Dialogs window. This action is equivalent to carrying out the TOOLS.DIALOGS_WINDOW command.
Parameters *x y*
Meaning - position where Dialogs window is to be open.
Type - integer
Default value - window being open at the recently saved position.

☑ *PASSWORDS*

Abbreviation - **PAS**
Operation type - opening the password manager window.

VARIABLE_DESCRIPTION variable_name[;window_parameters]

Abbreviation - **VAR**
Operation type - displays information about an indicated variable in the Variable Database.
Parameter *variable_name*
Meaning - declares an ASMEN variable, whose description included in the database is to be displayed.
Type - text
Parameter *window_parameters*
Meaning - declares a line name in the file SCHEMAT.TXT containing the list of fields, which are to be displayed.
Type - text

REMARKS *Optional "window_parameters" parameter must be separated with a semicolon from preceding ASMEN variable name.*

OPEN_MASK mask_name [opening_mode,x,y,monitor]

Abbreviation - **OP**
Operation type - an action to open any visualization diagram (synoptic or alarm). Additional parameter determines the procedure with previously opened diagram.
Parameter *mask_name*
Meaning - name of the diagram to open. The case of used characters isn't significant.
Type - text
Parameter *opening_mode*
Meaning - mode of a diagram to be opened
Type - symbol:

exchange - a current diagram shall be closed, then a new diagram will be open;

replace - like *exchange*, but new diagram will be opened physically in a place where the closed mask was placed (the x and y coordinate of the left upper mask corner will be identical);

group_exchange - all non-locked diagrams (which have permission to be closed) will be closed, and then a new diagram will be opened;

reduced_exchange - the same as for "group_exchange" mode, but the current diagram will not be closed, even if it is not locked;

new - opening a new diagram has no influence on the current diagram;

dialog - opening a new diagram does not influence the current one; after the diagram is opening, there will be no possibility of transition to any other diagram, until the diagram created by this action is closed;

temporary - opening a new diagram does not influence the current one; later transition onto another diagram will cause the automatic closing of temporary diagram.

Default value - opening mode in accordance with the mode given in the diagram definition

Parameters *x,y*

Meaning - place on the screen where the diagram has to be opened, determined as coordinates its left upper corner. No parameters or values of –1 mean that the diagram is to be opened according to its primary definition. The item applies in multi-monitor systems

Type - integer

Parameter *monitor*

Meaning - declaration of monitor where the diagram has to be opened (in a multi-monitor system). A value of 1 means a basic monitor, a value of 0 means that the diagram has to be opened on a monitor where the mouse is, a value of –1 means a use of a monitor resulting from the diagram definition and x and y coordinates. The item applies multi-monitor systems.

REMARKS *Attempt to open an already opened diagram causes its selection as the new current diagram. Rules of closing other diagrams remain unchanged.*

NOTICE When using mechanism of changing process variables during opening the mask, the *mask_name* parameter must be passed in accordance with one of the following syntax:

1. *mask_name#changes file*

In .VRT file the method of changing variable names should be declared. The file format is as follows:
line 1: *text description* – if the mask has its caption line, the text is entered into it;
line 2 to n: *source_variable_name;entered_variable_name*

2.
mask_name#source_variable_1;entered_variable_1[...#source_variable_n;entered_variable_n]

All pairs of changed variables are passed directly in the action content.

PANEL

Abbreviation - **PAN**
Operation type - selection of the Control Panel window

REMARKS *If the Panel window is minimized, it will be restored automatically.*

PAUSE [delay]

Abbreviation - **PAU**
Operation type - postpones execution of an action during given time period or until a key is pressed by an operator.

Parameters *delay*

Meaning - delay in seconds
Type - integer

Default value - waiting only for depressing a key

REMARKS *Action for creating the presentation. Used in action sets.*

PRINT_REPORT file moment operation [operator] [note]

Abbreviation - **PRI**
Operation type - making a specified report and its printout using a printer. The completed report will be saved on the disk or deleted.
Parameter *file*
Meaning - report definition file name (a full path)
Type - text
Parameter *moment*
Meaning - report beginning time in ASTEL language convention
Type - text
Parameter *operation*
Type - symbol (LEAVE or DELETE)
Meaning - shall the report remain or has it to be deleted
Default value - the report shall remain
Parameter *operator*
Meaning - operator's name (if it appears in the report definition)
Type - text
Default value - no name
Parameter *note*
Meaning - the note text (if it appears in the report definition)
Type - text
Default value - no note

SHOW_REPORT file moment operation [operator][note]

Abbreviation - **SHO**
Operation type - making a specified report and displaying it on the screen. The completed report will be saved on the disk or deleted.
Parameter *file*
Meaning - report definition file name (a full path)
Type - text
Parameter *moment*
Meaning - report beginning time, in ASTEL language convention
Type - text
Parameter *operation*
Type - symbol (LEAVE or DELETE)
Meaning - shall the report remain or has it to be deleted
Default value - the report shall remain
Parameter *operator*
Meaning - operator's name (if it appears in the report definition)
Type - text
Default value - no name
Parameter *note*
Meaning - the note text (if it appears in the report definition)
Type - text
Default value - no note

SIGNALS [group_identifier_list]

Abbreviation - **SIG**

Operation type - operation of the action is as follows:
- if the list of group identifiers is not declared, alarms from all groups are signalled by sound (default state after program startup);
- if the list of group identifiers is declared, only the alarms assigned to any group, idenrtifiers of which are declared, are signalled since action performance;

SYNCHRONIZE_ARCHIVE source_resource, target_resource

Abbreviation - **SYN**

Operation type - the action forces performance of SQL type archives (declared in the action content) synchronization by the ASPAD module.

CONNECT_ALARMS mask_name,resource_name,[computer_name],[connection_type],[number_of_days]

Abbreviation - **CON**

Operation type - action allows adding, to the diagram of historical alarms with a given name, an alarms source determined by means of the resource and computer names. As the diagram name may be given the character *. It means an operation on the first found diagram of historical alarms (the operation concerns always only one diagram). Searching the alarms source is executed in following sequence: direct connection to the alarm station, connection by means of a indirect server of historical alarms, connection to an existing file archive.

Parameter *mask_name*

Meaning - diagram, to which designated source of historical alarms has to be added.

Type - text

Parameterz *resource_name*

Meaning - name of resource to add

Type - text

Parameter *computer_name*

Meaning - name of a computer including the resource; when alarm source is unimportant, the'*' character should be passed.

Type - text

Parameter *connection_type*

Meaning - determine a way of connection execution. It may assume the values as below:
- lack of parameter or * - default procedure, in the first step a direct connection to the operator server is searched. If it fails then an access by means an indirect server is searched. If any access to the log is not yet obtained then it is checked if on the local disk, there is an old archive downloaded previously.
- indirect server name – an attempt to connect only by means of a designated indirect server is executed. If it

fails then it is checked if on the local disk, there is an old archive downloaded previously.
- NO_SERVER – a direct access to the operator server is only searched. After a failure an old local archive is displayed (if it exists).
- LOCAL – a connection to a local archive occurs; any attempts to search an alarms source in the network are not executed.

Type - text

Parameter *number_of_days*

Meaning - declares number of recent days, from which log files of alarms should be transferred. Lack of parameter means transfer of the complete log.

Type - text

NOTE *Action is performed only when the alarms system operates in the Historical_Viewer mode.*

ACKNOWLEDGE_ALARMS [alarm_number]

Abbreviation - **ACK**

Operation type - confirmation of all alarms showed on all open list diagrams or only one indicated by the *alarm_number* parameter.

Parameter *alarm_number*

Meaning - declares the number of alarm, which should be acknowledged (pending alarm only).

Type - integer

TRANSFER source_data_name target_data_name message_flag

Abbreviation - **TRA**

Operation type - carrying out of the transfer operation of an ASMEN variable to other variable. If variables are assigned to a real transmission channel then data transmission to or from the controller is performed.

Parameter *source_data_name*

Meaning - name of ASMEN variable, the value of which is being transferred

Type - text

Parameter *target_data_name*

Meaning - name of ASMEN variable, a value of the other variable is transferred to

Type - text

Parameter *message_flag*

Meaning - parameter *message_flag* is a number, which determines if the message about possible operation error should be displayed. A parameter value equal to 0 means that no message is displayed. This option may be useful e.g. in a situation of a group of TRANSFER operations called from a schedule when it is no possibility to provide any confirmation of messages by the operator.

Type - integer

REMARKS *The action that does not check the type compatibility of variables. It is assumed that the system designer is responsible for an application design. The only verified feature is size compatibility of transferred variables.*

☑ *ASK text*

Abbreviation - **ASK**
Operation type - the action causes displaying a dialog window with a given text and YES and NO buttons. If the operator click on NO button then the action set will be stopped
Parameters *text*
Meaning - text displayed in a window. In case of long texts one can split it on lines by inserting backslash '\ ' character. The definition length cannot exceed 200 characters.
Type - text

REMARKS *Provided for use in action sets.*

☑ *REPORTS [x] [y] [directory]*

Abbreviation - **REP**
Operation type - opening the report management window. The action is equivalent to the TOOLS.REPORTS_WINDOW command.
Parameters *x y*
Meaning - declare the position of the window is to be opened
Type - integer
Default value - open the window in a recently saved position
Parameter *directory*
Meaning - name of the working directory of the Reports module. It allows to distribute the reports definition among several directories and then to start up the system with selectively chosen reports set.
Default value - the working directory being established on the basis of the initialization file.

☑ *DRAW_TREND trend_symbol start_moment end_moment [print]*

Abbreviation - **DRA**
Operation type - displaying the trend window with an optional printout
Parameter *trend_symbol*
Meaning - a unique string of characters explicitly identifying the trend. It is attributed to each of trends in process of parameter setting.
Type - text
Parameter *start_moment*
Meaning - a trend start moment, in ASTEL language convention
Type - text
Parameter *end_moment*
Meaning - a trend end moment, in ASTEL language convention
Type - text
Parameter *print*
Meaning - depending on configuration in initialization file automatic dump of trend window will be performed or dump parameters window will be opened.
Type - symbol:

Print - print trend
Brak parametru - no printing

SCRIPT file_name,script_parameters

Abbreviation - **SCR**
Operation type - command to execute a script from a text file
Parameter *file_name*:
Meaning - name of file containing the script. If any access path is not given then the directories of diagram paths (MASK_PATH) are searched. If any file extension is not given then the **vbs** is added by default
Type - text
Parameter *script_parameters*:
Meaning - parameters transferred to the executed script

REMARKS *The **asix** system allows using the scripts written in Visual Basic Script (*.**vbs** files) or JavaScript (*.**js** files) languages.*

SET_BITS data_name value [mask]

Abbreviation - **SET**
Operation type - setting the value of the ASMEN process variable that causes transmission of data value to the controller (if the process variable is in the transmission channel). The action is oriented to digital control.
Parameter *data_name*
Meaning - name of the ASMEN variable to be set
Type - text
Parameter *value*
Meaning - value to send that can be earlier modified by parameter *mask*
Type - hexadecimal number
Parameter *mask*
Type - hexadecimal number
Meaning - a mask determining what bits of the variable are to be set or reset to zero. Using this mask causes that current value of the data is read in the first step. Then, only these bits are modified (according to the parameter *value*) which are set in the mask. Finally, such prepared value is sent back to ASMEN. The modification action can be presented as follows:
new = (old&(~mask)) | (value&mask)
The masking mechanism allows changing only some bits in the variable. The other bits preserve their previous values. Lack of parameter means that an actual variable value is not read. The data is being set directly according to the parameter *value*.

NOTICE SET_BITS action handles 32-bit variables.

SET_CHART variable_name

Abbreviation - SET_C

Operation type - setting the time of CHART object; action synchronizes all charts displayed on the current mask with the time transferred in a process variable. The variable should be of 32-bits type and the time transferred in it must have the form compatible with D format of NUMBER object (the number of seconds dating from 1.1.1970). It is compatible with the way of transmission of *Data* type fields by AsBase module. The action is dedicated mainly to control CHART objects generated on the basis of data stored in AsBase's database.

Parameter *variable_name*

Meaning - the name of Asmen's variable

Type - text

TABLE name[, mode, x, y, monitor]

Abbreviation - **TAB**

Operation type - opens a window of variables table.

Parameter *name*

Meaning - defines a name of a file containing the table definition

Type - text

Parameter *mode*

Meaning - defines a kind of operation, which should be executed
o*pen* - opening the table.
e*xchange* - opening the table together with closing earlier opened tables on the same monitor.
c*lose* - closing the table. If the parameter *name* is equal to *, then all open tables are closed.

Type - text

Parameters *x, y*

Meaning - coordinates of upper left corner of table window. No parameters or a value of -1 mean that the coordinates written in the table definition should be taken. The item applies to multi-monitor systems.

Type - integer

Default value - -1

Parameter *monitor*

Meaning - defines number of monitor where the table should be displayed (a value of 1 corresponds to a basic monitor). A special meaning has the value of 0. It means displaying the table on a monitor with current position of mouse cursor; parameter omitting or its value equal to–1 means using a monitor resulting from the table definition and x, y parameters. The item applies to multi-monitor systems.

Type - integer

Default value - -1

TRENDS [x] [y]

Abbreviation - **TRE**

Operation type - opening the window with the list of trends that can be displayed and printed but may not modified. This action permits to use trends protected against undesired changes that could be made by an operator.

Parameters *x y*

Meaning - window position

Type - decimal number
Default value - the window is centered on the screen

CLEAR_SOUND

Abbreviation - **CLE**
Operation type - clearing the sound of the alarm signal. In case of network stations the sound is cleared on all linked stations.

HIDE_ALL

Abbreviation - **HIDE**
Operation type - minimization of all open application windows to an icon on right part of the taskbar. As a result an automatic reduction of processor loading occurs. A return to the previous state is performed by double clicking the **asix** icon on the taskbar. The function of all windows hiding works also in system without any taskbar. In this case all windows are hidden, except the panel window or constructor window which are only minimized.

RUN command

Abbreviation - **RUN**
Operation type - execution of a program
Parameters *command*
Meaning - name of the program including parameters, which is to be run.
Type - text

SCREEN_PRINT mode[,printer_name]

Abbreviation - **SCRE**
Meaning - the action causes printout of all screen or active window; it is an equivalent of using the keyboard shortcut *Alt+F5*.
Parameter *mode*
Meaning - determines the object of printout.
Type - symbol
Window - operator action causes the active window printout;
Screen - operator action causes the screen printout.
Parameter *printer_name*
Meaning - when declaring the parameter, the printer name should be compatible with ones displayed by As program (e.g. in window of screen printout). Omission of this parameter means the printout on default system printer.

SEND_VALUE data_name value

Abbreviation - **SEN**
Operation type - setting values of the ASMEN process variable that causes transmission of data values to the controller (if the process variable is in the transmission channel). The action is oriented to controlling the analog value.

Parameter *data_name*
Meaning - name of the ASMEN variable to be set.
Type - text
Parameter *value*
Meaning - send value
Type - integer or floating-point number

EXCLUDE operation_type number

Abbreviation - **EXC**
Znaczenia - the action allows to change an exclusion state of a given alarm
Parameter *operation_type* - one of two values:
add - an alarm exclusion switching on
remove - exclusion removing
Parameter *number*
Meaning - may be a number of a single alarm or a name of alarms set file. One should give a file name without extension and access path – an extension **exl** is added automatically and files are searched in the working directory of alarms system.
An action „EXCLUDE add file_name" is equivalent to the operation *Add from file* executed in the exclusions support window.
An action „EXCLUDE remove file_name" is equivalent to the operation *Remove from file*.

PERFORM_INPUT [operation_range]

Abbreviation - **PE**
Operation type - sending the input operation completion signal. This action is used together with visualization objects. In a normal operating mode, the objects, after choosing an input value, do not carry out its immediate sending. They wait for the signal that is carried out by the PERFORM_INPUT action.
Parameter *operation_range*
Meaning - defines whether operation concerns the current diagram or all open diagrams
Type - symbol
current - signal to objects on the current diagram
all - signal to objects on all the open diagrams
Default value - signal sending to the objects on the current diagram only.

CLOSE_MASK [mask_name][monitor_number]

Abbreviation - **CLO**
Operation type - closing a current or declared diagram
Parameter *mask_name*
Meaning - contains the description of diagram names that are to be closed. Wildcard signs * and ? can be used to declare simultaneous closing of the whole diagram groups.
Type - text
Default value - closing a current diagram only
Parameter *monitor_number*

Meaning - determines a number of monitor the action refers to.

NOTICE *If explicitly any diagram name was not given then diagrams with set option „No close" will not be closed. If the parameter is an explicitly given diagram name then this diagram closes even if the option „No close" was set in its parameters.*

NEW_TIME

Abbreviation - **NEW**
Operation type - opening a dialog window allowing changing the system time. The action is equivalent to the command OPTIONS.TIME_CHANGE in the Control Panel and the Designer Window.

CHANGE_BITS data_name mask

Abbreviation - **CHA**
Operation type - changing the values of selected bit of the ASMEN process variable. The procedure of an action carrying out relies on reading a current value of a variable from the controller, inverting the defined bits and sending such prepared value back to the controller.
Parameter *data_name*
Meaning - name of the ASMEN variable to be set
Type - text
Parameter *maska*
Meaning - mask defining what bits of variable are to be changed (inverted). Modification action can be presented as follows:

new = (old^mask)

Type - hexadecimal number

NOTICE SET_BITS action handles 32-bit variables.

DISPLAY_FILE file_name

Abbreviation - **DIS**
Operation type - order to display a file
Parameter *file_name*
Meaning - name of the file to display
Type - text

17. Actions Scheduler

The actions scheduler is a module of the AS program that enables to execute the selected actions automatically. The time moment of execution of individual action can be defined following the two criteria:

system time	- actions are executed at the exact moment. Execution of actions can be restricted to the chosen day of month. The time, when the action starts running is given exact to a minute.
change of variable values	- actions will be executed when the given conditions for the process variable value occur.

Typical application of the scheduler is automatic periodical calculation of reports or execution of actions related to database service after a signal from a controller has received. There are no restriction related to the type of action that is to be executed by the action scheduler module. However, it should be remembered, that at the moment, when the necessity of action executing is realized the system operator's activities are unknown. For that reason execution of action that causes change of current window (e.g. OPEN_MASK) can be a kind of surprise for the operator.

Regardless to the chosen action activation method, there is possible to ask the operator to allow the given action to be executed. The mechanism causes, that before execution of an action, the dialog window describing the desired action will appear on the screen. Then the operator is able to lock the action execution.

17.1. Definition of Actions Schedules

There is the command in the menu of the Designer Window destined for declaration of actions executed by the Scheduler.

The time-depending actions are declared by the command TOOLS.ACTION_SCHEDULER > TIMES, called from Designer window, or with use of Architect program:

Architect > *Fields and Commands* > *Scripts and actions* module > *Scheduler* tab > ***Term*** parameter

The variable-depending actions are declared by the command TOOLS.ACTION_SCHEDULER > EVENTS, called form Designer window, or with use of Architect program:

Architect > *Fields and Commands* > *Scripts and actions* module > *Scheduler* tab > ***Event*** parameter

All the declarations introduced by the author of application are stored in the application configuration file.

17.1.1. Defining the Time-Depending Actions

The command TOOLS.ACTION_SCHEDULER.TIMES causes opening of the following dialog window that enables defining the scheduler's actions that start running with regard to time conditions.

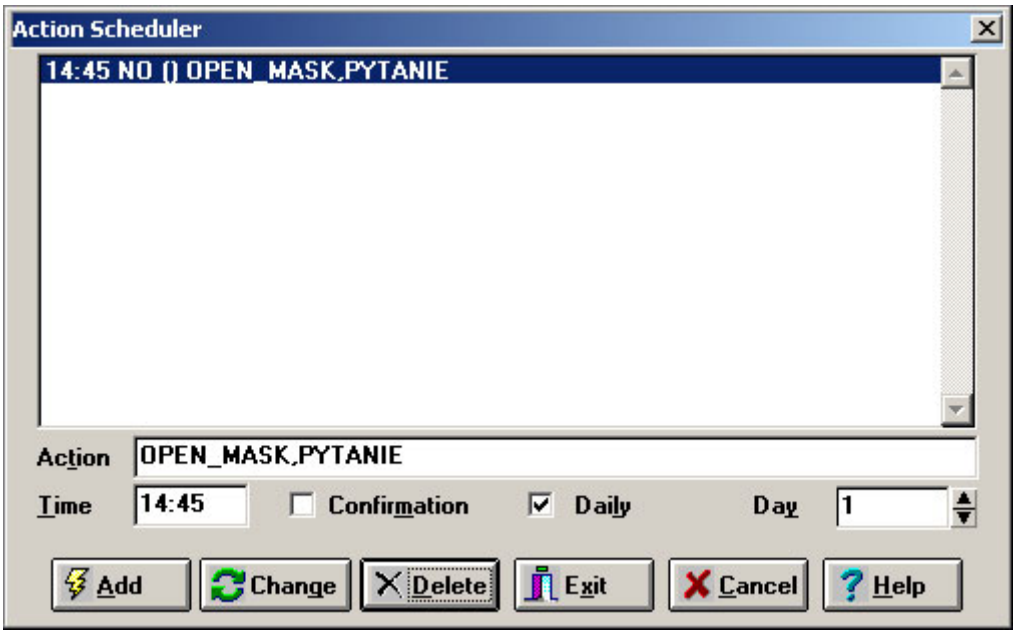


Figure 17-1. 'Action Scheduler' Window.

The window for event definitions consists of three parts. In the upper part of the window there is the list showing all the events that have already been defined. The form of description is the same as structure of the TERM item in the initialization file. In the middle part of the window there are fields used to define a new time-depending action or to change an existing one. The purpose of individual fields is as follows:

Action edit field - content of the action that is to be executed; if the content of the action written down is correct, the text in the field is automatically formatted;

Time edit field - the field used for time definition (exact to one minute), when the action is to be executed;

Confirmation checkbox - if selected, the operator will be requested to confirm execution of the action;

Daily checkbox - if selected, the action will be executed every day;

Day edit field - the field used for definition of the day of month, when the action is to be executed; the content of the field is meaningful only when the field *Every day* is not selected.

In the lower part of the window there is a set of buttons, clicking them causes the following operations execution:

Cancel button - closes the window and all the introduced changes are ignored;

Change button	- changes settings of the selected (highlighted) item of the list of time events;
Add button	- adds a new item to the list of time events based on the current contents of editable fields;
Delete button	- deletes the selected item from the list of time events;
Exit button	- closes the window, and all the currently defined time events will be stored in the application initialization file.

17.1.2. Defining the Event-Depending Actions

The command TOOLS.ACTION_SCHEDULER.EVENTS causes opening of the following dialog window that enables defining the scheduler's actions that start running with regard to monitoring of values of process variables.

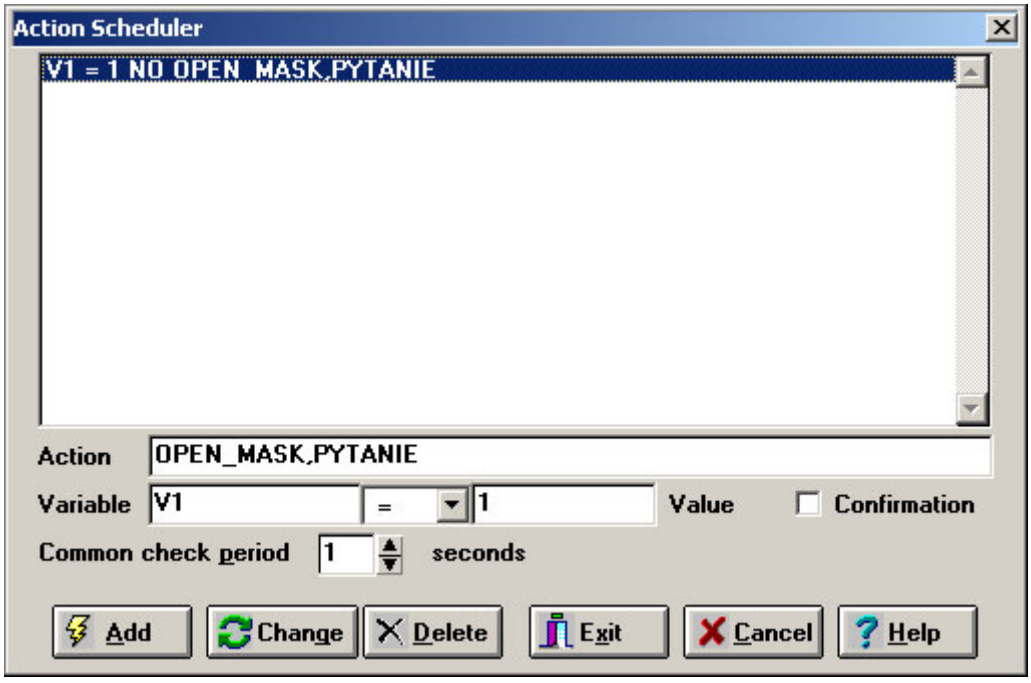


Figure 17-2. The Window for Event Schedule Definition.

The event definition window consists of three parts. In the upper part of the window there is the list showing all the events that have already been defined. The form of description is the same as the EVENT item structure in the initialization file. In the middle part of the window there are fields used to define a new event-depending action or to change an existing one. The purpose of individual fields is as follows:

Action edit field	- contents of the action that is to be executed; if the content of the action written down is correct, the text in the field is automatically formatted;
Variable edit field -	- name of the process variable, that determines execution of the action;

field of condition - the drop-down list field used for selection of the condition to check; there are four conditions available:
= - the variable is equal to the given value;
> - the variable is greater than the given value;
< - the variable is less than the given value;
& - bitwise conjunction of the variable and the given value;

Value edit field - the number used to compare with the monitored process variable; it is the integer or floating-point number; the program automatically matches types of variable and value; the condition of "&" type cannot be used for floating-point data;

Confirmation checkbox - if selected, the operator will be requested to confirm execution of the action;

Common check period edit field - the field that is referred to all the items simultaneously. It defines the time period between the two consecutive checking of values of the monitored variables.

In the lower part of the window there is a set of buttons. The purpose of them is the same as in the time event definition window.

Choosing the appropriate updating time is essential. When the condition of the action has been met, the consecutive action execution will be possible only after that, when situation that the condition is not met would be detected (the soonest time is the next checking cycle). As a result, the soonest time of repeating the action is twice longer than conditions checking period.

The updating time should not also be longer than the updating time defined in the ASMEN manager (lack of appropriate frequency of reading may withhold input of variables).

17.1.3. Storage of Parameters

After the EXIT button of the defining window would have been used all the currently defined actions are written to the SCHEDULER section of the initialization file. It is not equivalent to the final saving of the new events. The AS32 program will use the new events, but in order to save them on the disk an execution of the writing operation of the application initialization file is necessary (the command FILE.SAVE must be executed or the AS32 program must exit saving the initialization file).

17.1.4. Execution of Time-Depending Actions

When the AS32 program starts in the application mode the monitoring of time and variable values begins. Thus the moments of time-depending actions execution can be detected. If conditions that force action execution will be met, the attribute of action execution confirmation is checked. If the action has been defined without setting that

attribute, it is immediately executed. Whether confirmation is required, the following dialog box is displayed to inquiry for the right to execute the action.

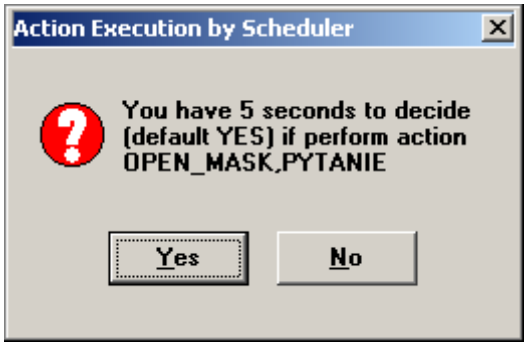


Figure 17-3. 'Action Execution by Scheduler' Window.

The operator can enable or disable the action execution. No reaction of the operator causes that the window will be closed within about 5 seconds and the action will be executed.

If the operator disables execution of the action, the attempts to execute it will be no longer undertaken.

18. Objects of the asix System

The **asix** system is provided with a set of objects that enable designing synoptic diagrams for visual presentation of technological systems. You can create the diagrams when AS32 program runs in the **Designer** mode. This mode provides the function of inserting, defining and modifying the objects on the diagrams.

An object is a kind of structure, which can be described with the **set of parameters** assigned to it. The parameters define the object attributes (dimension, color and fonts), relations to variables of the **asix** system, way of reaction to changes of system variables, operator's actions, etc. The object parameters are defined with use of the object definition window. The object is able to present itself, i.e. to be displayed on the diagram. The displayed **image** of the object depends on its **state**. The objects are prompted to be displayed by means of the internal **refresh signal**. The operator's action can also make changes of the object state.

18.1. Set of Objects of the asix System

The standard set of objects of the **asix** system includes:

ASBASE - provides communication with AsBase database
BAR - displays analog value in the form of a bar
BUTTON - button with capability of performing action or control
CALCULATOR - performs calculations on variables
DATE +TIME - displays date and/or time
ELLIPSE - draws ellipse – static object
ELLIPSES - ellipse reacting to the variable's status
EXPRESSION - displays result of calculations performed on the variables
LINE - draws line – static object
LINE HV - line perpendicular to the diagram's edge – static object
LINES - displays lines depending on the status of variable
LINES HV - displays perpendicular lines depending on the status of variable
MESSAGES - displays one of many specified texts
MOTOR - displays drive status with possibility of control (table)
MOVE CONTROLLER - allows for animation the movement of technological mask objects
NUMBER - displays variable value
STRING - displays the string of characters
OBJECTS CONTROLLER - allows for visibility control of the technological mask objects
PICTURE - displays the bitmap– static object
PICTURES - displays bitmaps depending on the status of variable
PIPELINE - draws pipeline segments – static object
POINTER - displays analog value as a circular indicator
POLYGON - draws any polygon – static object
POLYGON HV - draws polygon with perpendicular sides –static object

POLYGONS	- displays one of polygons, depending on the v variables status
POLYGONS HV	- displays one of rectangular polygons, depending on the variable value
POLYLINE	- draws segmented line– static object
POLYLINE HV	- draws segmented rectangular line– static object
POLYLINES	- displays segmented line depending on the variable status
POLYLINES HV	- displays rectangular segmented line depending on the variable's status
PRESENTER	- signals opening selected diagram
RECTANGLE	- draws rectangle –static object
RECTANGLES	- displays rectangles depending on the status of variable
REFRESH TEST	- tests the speed of refreshing the diagram
REPORT	- calculates and displays previously defined report
SELECTOR	- enables controlling the variable operating the description texts
SLIDER	- bar indicator with value setting
SWITCH	- sends two different controls depending on the variables's status
SWITCH SET	- set of control switches
SYNCHRONIZER	- signals performing the „send controls" action
TEXT	- writes text with any font – static object
TEXTS	- displays one from many texts depending on the variable status
WORK POINT	- displayes cursor in X ,Y variables status
CHART	- displays archival data as a chart

The above objects will be described in the later. Description of the object will contain, in every case, the example window defining the given object and list of the object parameters. Moreover, in most cases, description will include the window with the hypothetical object or with the set of objects of the given class.

For majority of objects their dimensions can be freely defined by use of the mouse (in the same way as any window is resized the corner of the object can be "dragged" with a mouse). Dimensions of some objects (all the bitmaps) are fixed and cannot be changed. Sometimes an object cannot be resized if scaling would lead to lost of data that it contains or dimensions of the object may depend on the method of setting parameters. The more detailed information about this subject has been distinguished in description of every object.

Definitions of objects that create synoptic diagrams are stored on a disk in binary form (files **.msk*). Objects of newer versions automatically carry out conversion of definitions of the older objects whether the last ones are being read from a disk. Every time execution of diagram definition saving on a disk saves the converted definitions.

18.2. Glossary

Communication Error
Measurement Error
Control Data
Monitored Data
Measured Data
Controlled Data
Dynamic Object
Static Object
Selectable Object
System Variable

Communication Error

The communication error may occur while data are transmitted between the controller and the computer where the **asix** system has been installed. Every dynamic object can respond to the communication error by changing its image.

Measurement Error

The measurement error may occur between the measuring sensor and the object or in the sensor itself. Sometimes the sensor is able to detect the measurement error (it needs the appropriate software of the controller) and pass information about it to the object by means of the control data. Some dynamic objects can response to the measurement error by changing its image.

Control Data

The control data is a kind of the monitored data. It plays the supplementary role for each object that uses measured data. The convention has been adopted that the logic product (conjunction) of the given control data and the associated state, whether different from zero, forces the object to indicate the measurement error. The appropriate software of the controller is necessary for to work out the control data.

Monitored Data

The monitored data is the system variable that is being read by a dynamic object during refreshment process. The monitored data are received from the controller. Change of value of monitored data may cause change of the object image.

Measured Data

The measured data is the system variable that associated with the analog measured value. The data are received from the controller and is being read by the dynamic object while refreshed. The measured data is the subclass of the monitored data.

Control Data

The control data is the system variable that can be set by operator and sent to the controller. An object writes the data. In some cases the variable can be read from the controller before writing due to write to the controller the logic sum of the value been read with the written field set to zero and the written value. This type of writing can be executed if the parameter "read first" is set for the controlled data. The monitored data can be the control data as well.

Dynamic Object

The dynamic object is one that changes its state (and image) when the diagram is refreshed. The state can be changed when the system variable changes its value or when time expires (or in result of operator's action). Parameters of dynamic objects usually include system variables.

Static Object

The static object is one that does not change its state when the diagram is refreshed. Static objects are not related to system variables. Set of parameters of such an object is restricted to the attributes of the image of the object.

Selectable Object

The selectable object is one that changes its status (and image) when the diagram is refreshed. The change of status is due to operator's action. These objects are usually used for sending controls (they store their values in system variables). Such an object should react to the operator's choice. As response to the choice the object should:

- change its state and redraw itself or
- enable operator to enter values (or to review possible states of the those object, that any values have already been assigned to them) and
- store the new value in the system variable and, initialize the process of sending of it to the PLC.

The selectable object can be classified as well the dynamic one.

System Variable

The system variable is one defined in the **asix** system. It has its symbolic name, type and status. It can store values that have been read from the controller, values that are to be sent to the controller or other values that are stored e.g. for needs of objects.

18.3. Blinking Attribute

Following objects of the **asix** system have the possibility of declaring the blinking attribute when setting the mode (color) of display:
BAR
NUMBER
MESSAGES
POINTER
EXPRESSION
TEXTS
ELLIPSES
PICTURES
RECTANGLES
LINES, LINES_HV
POLYLINES, POLYLINES_HV
POLYGONS, POLYGONS_HV

According to the above, it is possible to declare the color of objects displayed in two ways: with the pointer of the basic color set, in the shape of rectangle or through the blinking attribute setting indicator (in the shape of round indicator):

If the color of blinking attribute setting indicator is ashen, this attribute has not been set and the object will be displayed in constant color, if the indicator is displayed in yellow or red, in turns with ashen, object's blinking has been declared. Indicator blinking in yellow means selecting one of the xor types, in red - means other blinking settings.

18.3.1. Blinking Declaration

To configure object to be displayed with blinking attribute, you should click on the blinking attribute indicator, which will open the window in which you should declare the mode and blinking colors. Following options are possible:
no action – constant colors;
xor mark background – background and mark colors blink, changing into opposite colors;
xor mark – only the mark color changes into opposite;
xor background – the background color changes into opposite;
color mark background – background and mark colors change into declared alternative colors;
color mark – mark color changes into declared alternative color;
color background – background color changes into declared alternative color.

The *Example* field shows, what will be the color layout during application run.

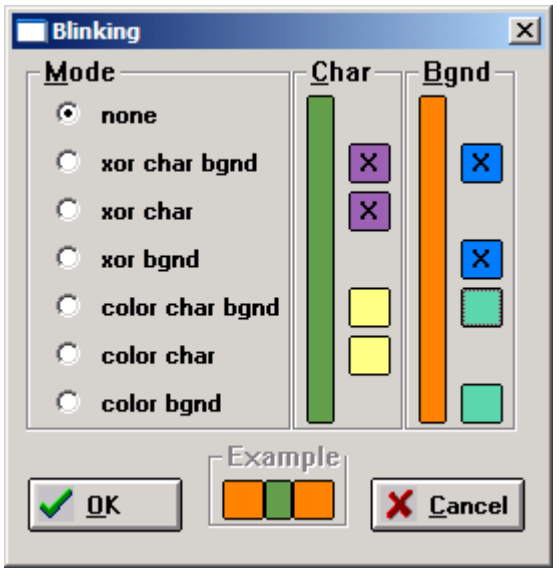


Figure 18-1. 'Blinking' Window.

The fields denoted by a green and orange colour are assigned to a basic color definition.

In the field and the opposite colours are displayed (xor of a basic color).
In the field and the alternative color are displayed.

18.3.2. Setting Blinking Frequency

To change the default blinking period of the objects with blinking attribute declared (333 ms) in the application configuration file the ***Blinking period*** parameter should be set up with use of Architect program:

Architect > *Fields and Computers* > *Masks* module > *Mics* tab

18.4. Objects

18.4.1. ASBASE Object

Object Parameters
Data Filtering
Archive set definition

The ASBASE object provides communication with an AsBase database, making it possible to browse its tables. It is a „transparent" object, put on an **asix**'s mask background. If global access to a database is needed, the object can be placed on the mask being displayed during all the time of an application running (e.g. menu). At the mask start the object operation consists in creation of connection with a proper data source, and at the mask close it consists in its closing (AsBase must be running during displaying the mask).

Browsing a single table takes place within the connection that has its unique identifier, defined during the ASBASE object configuration. Afterwards this identifier is used as the parameter of the ASBASE,*Navigate* operator action to determine the connection this action refers to. Besides the connection identifier definition, the ASBASE object allows to define the type of browsing table, table identifier, set of process variables the values from table records will be transferred to, and filter that determines which of records stored in a table has to be displayed on process masks.

The ASBASE object allows to browse the following types of tables: *archive sets*, *recipe groups* and *load history*. The way of filter definition is described below.

When opening the mask, which the ASBASE object is put on, it occurs: opening the connection with a given identifier, opening a specified table and transferring the individual field values of the last record in the table, which is compatible with a filter definition, to process variables of the variable set with a given identifier.

For graphic presentation of values of database record fields, NUMBER and STRING objects are the best suitable.

Data source record navigation is realized by means of the BUTTON object, which the properly configured ASBASE,*Navigate* action should be assigned to (see: *16.2. Description of Actions,* ASBASE action).

Parameters

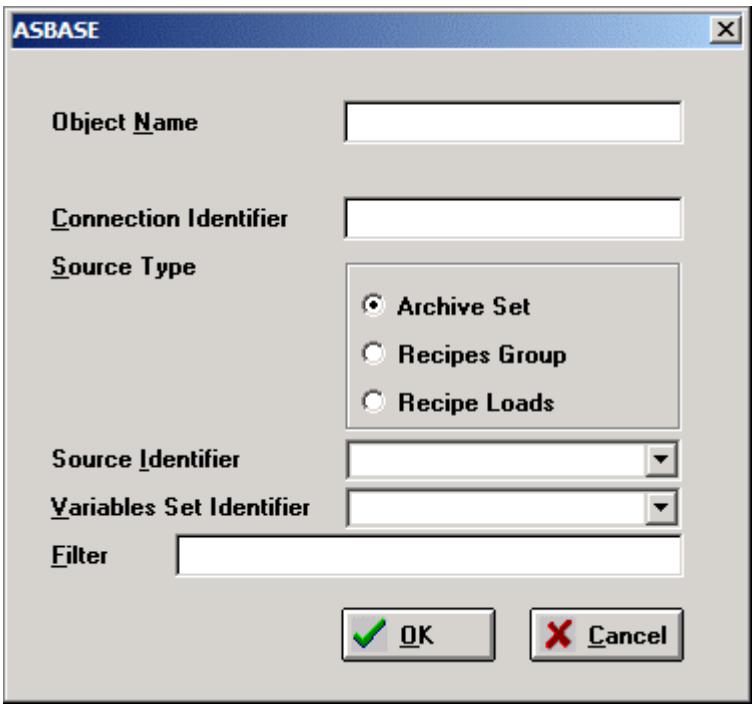


Figure 18-2. The AsBase Object Parameterization Window.

Object Name - a text box used to entering the optional object name. This name appears on the object list. If no name is declared, the object type and coordinates will appear on the list of objects.

Connection Identifier - a text box used to entering the identifier of connection with a given data source and variable set; it is used in the ASBASE,*Navigate* action, which enables navigation of records of given data source variables (see: *16.2. Description of Actions,* ASBASE action).

Source Type - a radio button provided to declare the data source type available in the AsBase module:
- *Archive Set*,
- *Recipes Group*,
- *Recipe Loads*.

Source Identifier - a combo box provided to choose the following identifiers, successively for individual source types: identifier of archive set, recipes group and recipe/recipe group loads.

Variables Set Identifier - a combo box provided to make it possible to choose a set of variable, by means of which values of demanded AsBase database fileds will be transferred to the **asix**'s system application (and displayed on a mask that is being designed).

Filter - a parameter that allows to select displayed variable values.

Data Filtering

Table browsing may be limited to records that fulfil criteria by means of data filtering. The filter may be defined during the ASBASE object configuration and by the ASBASE,*Navigate* operator action (see: *16.2. Description of Actions,* ASBASE action).

The filter syntax corresponds to the WHERE clause of SQL with extensions that allow to isert process variable values and predefined filters into this clause. The WHERE word should not be passed in a filter definition. The filter insert is put in braces.

The extended insert that allows to use a process variable value has the following syntax:

{V:*variable_name*}

A variable value is put into an insert. If the variable has a number value, the value is changed into text.

The insert that defines elements of a time filter has the following syntax:

{T:*code*[:*field_identifier*]}

where:

code:

t - today
y - yesterday
sy - since yesterday
tw - this week
lw - last week
slw - since last week
tm - this month
lm - last month
slm - since last month
ty - this year
ly - last year
sly - since last year

field_identifier - an optional table filed identifier which a time filter is to be applied to. If it is missed, it is taken that the filter concerns the time of record write into a table. If a field identifier is passed, the field should be of *Data* type. Use of this insert causes that all insert text will be replaced by SQL expression limiting a set of table records to ones which time has a value from time interval defined by *code*. In this way, it is possible to display table records concerning only the previous day (*y* code), for example. The filter is not modified during the connection time, even when a given time interval has expired, i.e. if *today* is declared and the connection is opened for a second day, there will be the records from the last day displayed. Not before the connection is opened again or the same filter is reused in the operator action, the data will be updated. The field identifier is created by adding the 'V_' prefix to the field name.

EXAMPLES

The {T:t} filter causes displaying only the records written during the day of setting the filter.

The {T:y:V_field1} filter causes displaying only the records with the time type field field1 the time of which is from the *yesterday* time interval.

NOTICE Inserts may not occur within filter elements put in square brackets [] and quotation marks (` and "). To put an insert content between this limiters, it is necessary to put the limiters in braces.

EXAMPLE: {`}{V:*variable_name*}{`}

The above part of a filter will be replaced by a variable value put in single quotation marks (`).

Besides table field identifiers, it is also possible to use identifiers concerning constant table fields, i.e. fields which always occur in a table apart from fields of an archive set or recipes. There are fields like: Time, UTC Time, Source, etc. - there are names of the columns which appear in the AsBase module during archive set/recipes/recipe load history browsing. These fields may also be elements of a filter. Identifiers of these fields are names of table columns – in the form defined in the database („V_" prefix is not added).

Table 18-1. Identifiers of Constant Fields of AsBase Database Tables.

Identifier	Name of AsBase Module Column
LOCALTIME	Time
UTCTIME	UTC Time
STATUS	Status
SOURCE	Source
NAME	Name
LOADBY	User
TARGET	Target
CREATEDBY	Author

As mentioned above, to use archive set/recipes/recipe load history as a filter element, the name of this field should be preceded by the "V" prefix (besides constant table fields). In case of archive set, there is a value status and value time stamp connected with each field value. To use this elements, it is necessary to preceed a field name of archive set by the „S" or „T" prefix, properly.

AN EXAMPLARY USE OF THE FILTER IN THE ASBASE OPERATOR ACTION

ASBASE,NAVIGATE,conMie\Filter\v_id_miesz like {`}{V:1PD_id_m1}{`} OR v_id_miesz like {`}{V:1PD_id_m2}{`} AND {`}{V:1PD_id_m2}{`
}<>"

The action sets the filter on the `conMie` link in the following way:

the value of the `v_id_miesz` field is equal to one of composition identifiers: 1 composition id (variable `1PD_id_m1`), 2 composition id (variable `1PD_id_m2`). Records with empty id of 2nd composition are excluded.

Archive Set Definition

To make it possible to display values of individual table fields, the names of process variables of NONE channel have to be assigned to these fields. It can be made by creation of a proper archive set in the AsBase module. The identifier of this set is later passed as an ASBASE object parameter. By default, constant table fields are not displayed in such a set. To assigne variables to constant table fields, it is necessary to choose the option *Show all fields* (from context-sensitive menu opened by clicking on the headings of table columns on the *Variables set – Fields* tab).

EXAMPLE

Below you can look at the definition of an examplary variable set. The constant field named Record count allows to assign the quantity of records that fulfil a filter criterion to a variable. Record counting occurs when establishing a connection, defining a new filter and the ASBASE,*Navigate* operator action (with the COUNT command). The value of ID is a consecutive number of a database record.

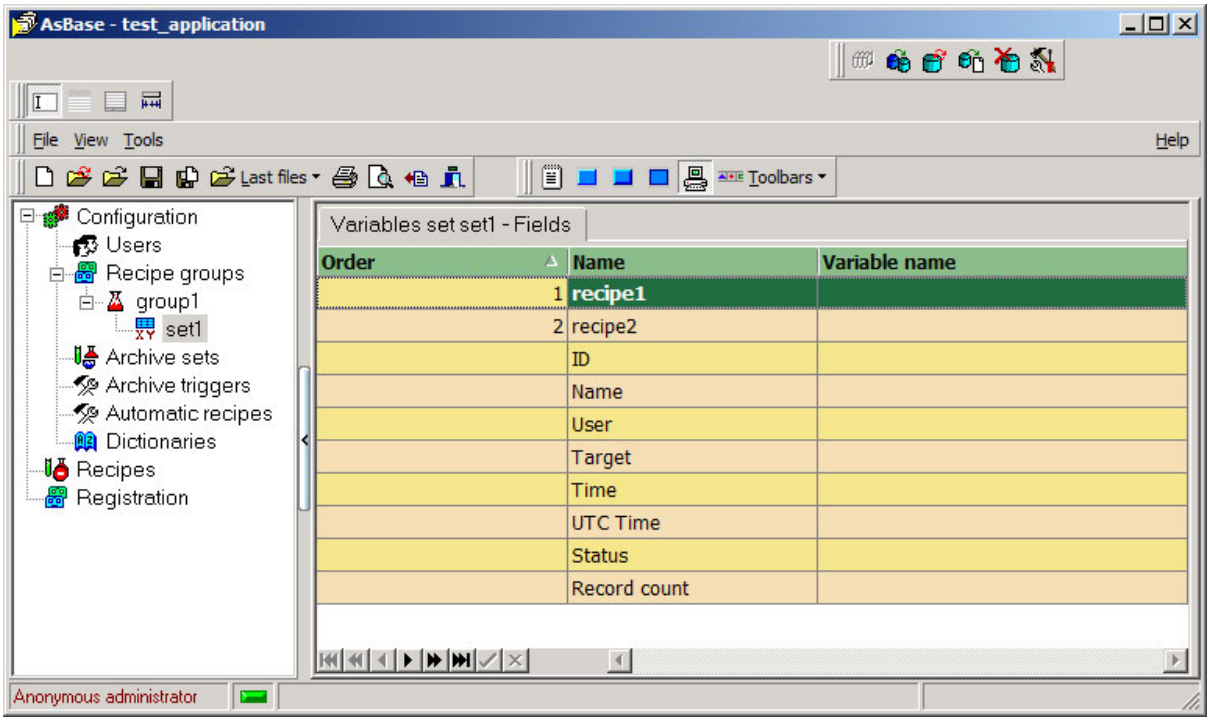


Figure 18-3. The Exemplary Definition of Variable Set in the AsBase Module.

18.4.2. BAR Object

Dimensions
Object Parameters
Object Name
Colors
Ranges
Frame
Direction
Variable
Contour
Configuring from the VarDef

BAR object enables displaying the bar shape on the mask that represents the system variable. It is a dynamical object. BAR may change colors and have markers assigned to it, which correspond to specific constant or variable values; it is possible to declare the blinking attribute for the status of defined limits exceedance. It signals measurement and communication errors. Using the contour in the format of black-and-white bitmap (from **asix** bitmap set or in the separate file), it is possible to freely shape the bar indicator.

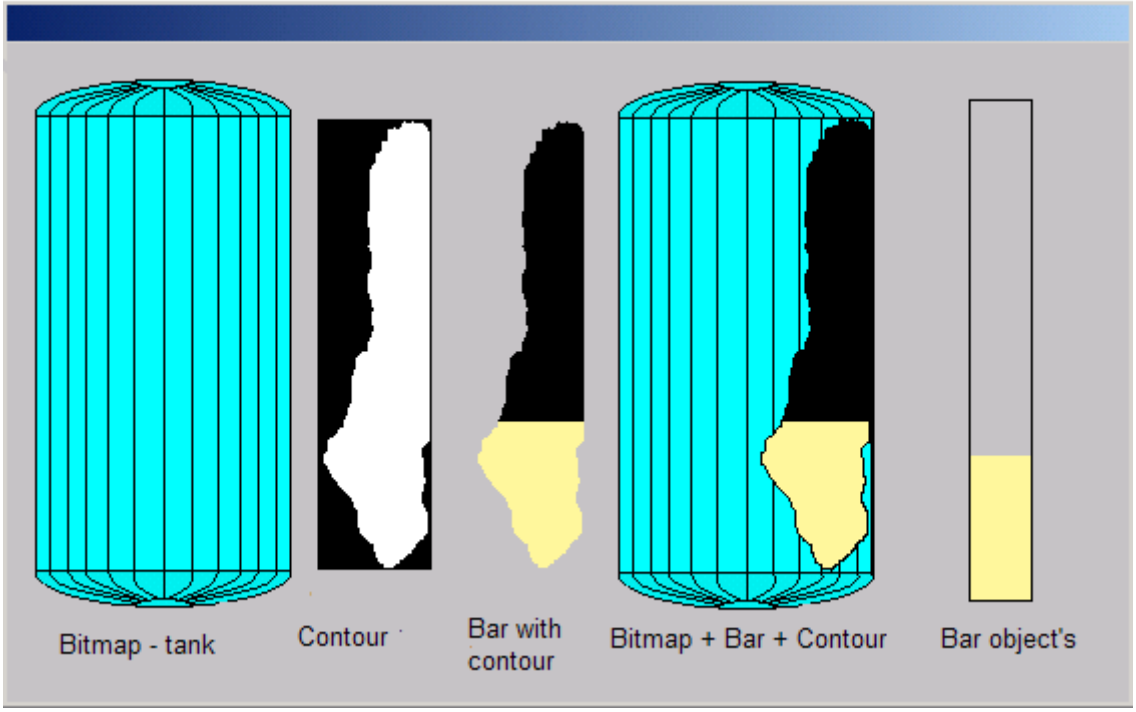


Figure 18-4. An Example of BAR.

Dimensions

Dimensions of BAR object can be defined with the use of mouse, or are determined by the bitmap dimensions, in case of using the contour for shaping the bar indicator.

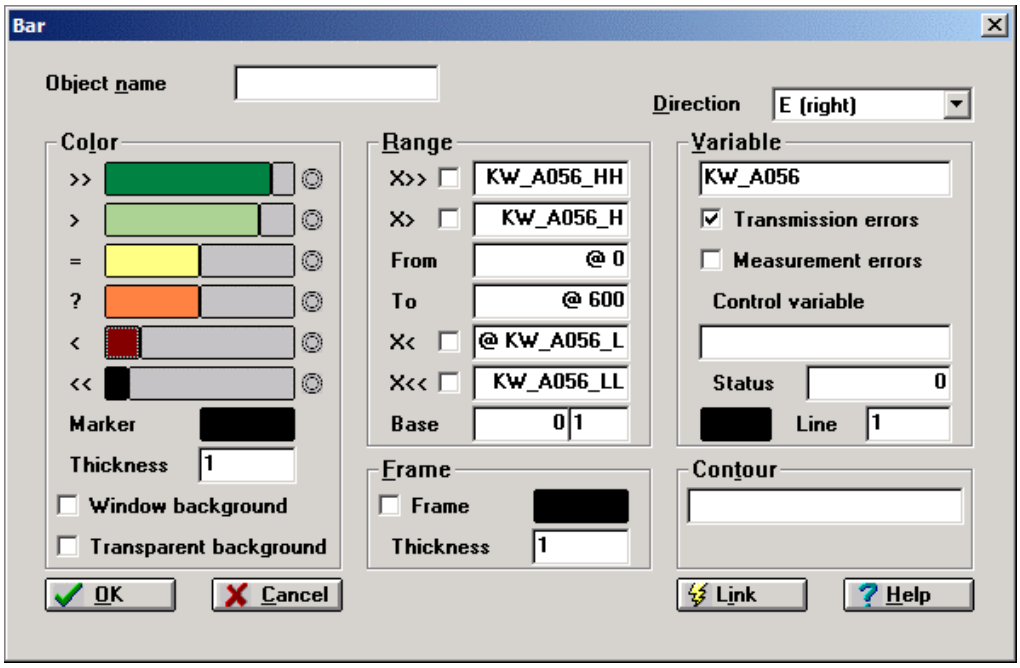


Figure 18-5. The BAR Object Parameterization Window.

Parameters

Object Name - this text box is used to enter the optional name of the object. if the name of the object will not be defined, its type together with coordinates will appear on the list of objects.

Color ->> - color fields enabling setting the bar color (left field) and the background color (right field) in case of exceeding the upper alarm limit. The indicator to the right from the color fields enables setting the blinking attribute.

Color -> - color fields enabling setting the bar color (left field) and the background color (right field) in case of exceeding the upper warning limit. The indicator to the right from the color fields enables setting the blinking attribute.

Color -= - color fields enabling setting the basic bar color (left field) and the background color (right field) of displayed number. The indicator to the right from the color fields enables setting the blinking attribute.

Color -? - color fields enabling setting the bar color (left field) and the background color (right field) in case of an error. The indicator to the right from the color fields enables setting the blinking attribute.

Color -< - color fields enabling setting the bar color (left field) and the background color (right field) in case of exceeding the lower warning limit. The indicator to the right from the color fields enables setting the blinking attribute.

Color -<< - color fields enabling setting the bar color (left field) and the background color (right field) in case of exceeding the lower alarm limit. The indicator to the right from the color fields enables setting the blinking attribute.

Color -| - a color box provided to specify the color of the markers.

Color - Window background **-** a check box provided to force the background color being the same as the background window color.

Color - Transparent background **-** a check box provided to force the t background.

Range -X>> - a check box and a numerical-text box provided to specify either a numerical value of the upper alarm limit or a name of the variable, which contains this limit. The default value is 90. When the variable name is to be specified, its logic condition may be additionally specified in the box situated rightward. See the next sections for details on the conditions. The check box forces drawing of a marker.

Range -X> - a check box and a numerical-text box provided to specify either a numerical value of the upper warning limit or a name of the variable, which contains this limit. The default value is 75. When the variable name is to be specified, its logic condition may be additionally specified in the box situated rightward. See the next sections for details on the conditions. The check box forces to draw a marker.

Range -from - a numerical-text box provided to specify either a numerical value or a name of the variable corresponding to the lower range of the bar. The default value is 0. The value may be potentially set from the variables database VarDef ('BarRangeFrom' field).

Range -to - a numerical-text box provided o specify either a numerical value or a name of the variable corresponding to the upper range of the bar. The default value is 100. The value may be potentially set from the variables database VarDef ('BarRangeTo' field).

R*ange -X<* - a check box and a numerical-text box provided to specify either a numerical value of the lower warning limit or a name of the variable which contains this limit. The default value is 25. When the variable name is to be specified, its logic condition may be additionally specified in the box situated rightward. See the next sections for details on the conditions. The check box forces drawing of a marker.

Range -X<< - a check box and a numerical-text box provided to specify either a numerical value of the lower alarm limit or a name of the variable

which contains this limit. The default value is 10. When the variable name is to be specified, its logic condition may be additionally specified in the box situated rightward. See the next sections for details on the conditions. The check box forces drawing of a marker.

Range - Base - a numerical-text box provided to specify either a numerical value or a name of the variable corresponding to the reference base of the bar. The default value is 0.

Frame - Frame **-** a check box and a color box provided to specify whether the bar should be provided with a frame and to specify the color of the frame.

Frame - Thickness **-** number field enabling specifying the border thickness (1 by default).

Direction - a combo box provided to determine the direction of the bar growth. The options are: N (up), E (right), S (down), and W (left). The default option is E.

Variable - a text box provided to specify a name of the measuring variable. Specifying the name is obligatory. By clicking the right mouse button on this box you can open a window containing the list of available variables to make selection.

Transmission error **-** a check box provided to warn about transmission errors. When they occur, the bar is crossed along the line of specified parameters. Default option.

Measurement Errors **-** a check box provided o warn about measurement errors. When they occur, the bar is crossed with the lines of specified parameters. It requires specifying the ***Control Variable***.

Control Variable **-** text field allowing specifying the name of control data. Specifying it is obligatory, if the ***Measurements errors*** option has been selected. By pressing the right mouse button in this field a window containing the list of available variables can be opened, and you can make a choice in it. It is possible to define the name of control variable using notation with **#** character. If in the field of variable name the #_ characters with suffix are entered, then during the application's run it will be treated as the variable with name containing the name of monitored variable instead of the # character.

Status - a text box provided to insert a hexadecimal mask. If the logical product of this mask and the control variable is different from zero then a measurement error occurs.

Line - a color box of the line, which crosses the object when errors occur and a numerical box, which used to determine the line thickness. Selection of the color box causes displaying of the color selection window.

Contour - introducing the masking contour in the BAR object enables visualization of level changes in tanks of any shape (conal, spherical etc)
The masking contour should be prepared with any graphics editor supporting BMP format. The contour must be monochrome (black-and-white), of wich white is „transparent" for the bar, and black for the object's background.

NOTICE Empty texts and the texts consisted of white spaces are acceptable in limit fields and using them is treated as abandoning verification.

The displayed bar may change its color either on exceeding limits or meeting certain logic conditions. Exceeding can be specified as either the numbers (fixed) or the system variable names. If the exceeding is specified as a system variable name, a logic condition to be met can be specified in the box close to the name (rightwards).

The logic condition is a sequence consisting of a „&" or „|" character followed by a hexadecimal number (condition). With the „&" character, condition is met if the logic product of the variable and the condition equals to the condition whilst with the „|" character, condition is met if the logic product of the variable and the condition is different from zero. In other words, in the first case, to change the attribute it is necessary to have all the bits set in the given variable whilst in the second case at least one bit of the condition must be 1. When specifying the condition, the „|" character may be neglected.

Moreover, markers (dashes) may be inserted in the places corresponding to the limit exceeding. The markers are to be drawn after setting „yes" in the check box close to the box where the limit exceeding are written in. The default reference level is a minimum level (the level „from" equals to the „base" level). If the reference level i.e. the so called „base" is different from the minimum value, the bar does not begin from the object base. This mode is used to display gradients. The minimum level should be different from that maximum (Note: it must not be greater!). The reference level should be contained within a range between the minimum and maximum levels. During the refresh operation (provided that the color change is not necessary) the bar gradient corresponding to the object measurement variable is drawn only. When a transmission error occurs, the object is crossed transversely along the direction of bar movement with the lines having the declared parameters. When a measurement error occurs, the bar is crossed with the transverse lines having the parameters as declared above. Additionally, when one error occurs (or two errors occur at the same time) the bar may change its colors. The bar may be surrounded with a frame.

Configuring from the Variable Definitions Database

BAR object is additionally provided with features, enabling setting the following parameters from the variables base:

- range of variation e.g. 0, 100 according to the displayed variable's name,

- limits, on exceeding of which the change of color will occur, e.g. the color of name of **Limit**, **Limit&10** variables **(**variable name with the bit mask**)** according to the displayed variable's name,
- name and mask of the control data.

In this purpose, the number object has been additionally provided with ***Link*** button, pressing of which causes inserting the contents of given field in the database, preceded with **@** character, into every position, which may be potentially set from the variables base.

In the example below, pressing the Link button has caused inserting the contents of fields from the base for:

- critical maximum limit – variable named KW_A104_HH
- maximum limit – variable named KW_A104_H
- minimum limit – variable named KW_A104_L
- critical minimum limit – variable named KW_A104_LL
- control data – variable named KW_A104_S
- control data's mask – 1

18.4.3. BUTTON Object

Dimensions
Object Parameters
Object Name
Action
Text
Picture
Description
Code
Justifying
Sending
Button
Get a Key

The BUTTON object permits the operator to perform a certain action by clicking a button. This is a selectable object. Action starts after the button is selected either by clicking the mouse button or after pressing *Enter*. The action can also be carried out by pressing the keyboard shortcut. Execution of an action may be password-protected. The object does not depend on the variable states but changes appearance in function of the button pressed. It can be transparent and multiple overlaid as well.

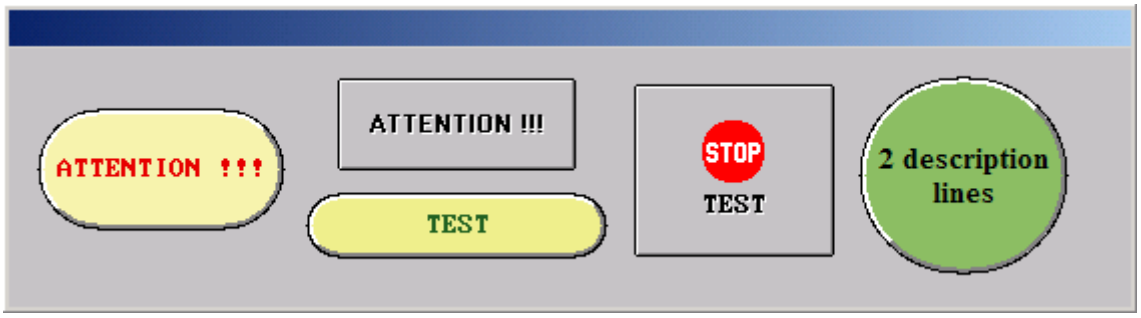


Figure 18-6. The BUTTON Example.

Dimensions

The minimum dimensions of the BUTTON object are determined in function of its caption, fonts and size of possible bitmaps written in the button. BUTTON may be enlarged using the mouse. Trying to minimize the object below its natural size will fail.

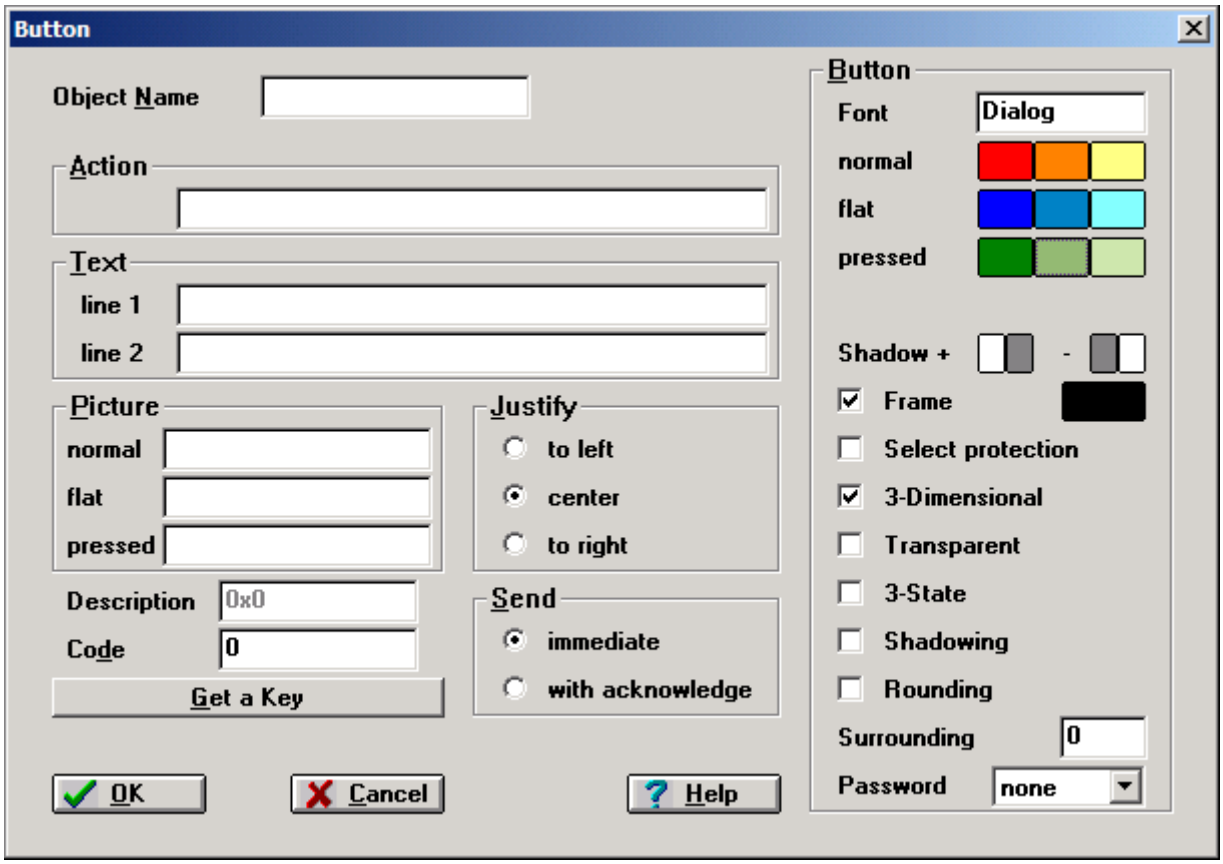


Figure 18-7. The BUTTON Object Parameterization Window.

The button code (the scan code) can be either written in the "Code" box or, simpler, by clicking simultaneously appropriate keys after selecting the "Get a Key" function. Inside the button a caption along with a picture (bitmap) may exist in parallel. Different bitmaps can be used depending on the button state

(pressed, released). A transparent button may be overlaid onto other objects. The objects overlaid by a transparent *BUTTON* can be static or dynamic types.

The button can function in either an immediate mode or with acknowledge mode. In the first case, the action starts in the moment of release of the mouse button. In the second case, the button remains pressed till the moment of either acknowledge of the action by means of the *Grey+* key or performing of the adequate action.

Parameters

Object Name **-** this text box is used to enter into it the optional name of the object. If the name of the object will not be defined, its type together with coordinates will appear on the list of objects.

Action **-** a text box containing the text of the operator's action to be made after pressing the button (see description of operator's actions). It may be empty.

Text-line1 **-** a text box containing caption of the button.

Text-line2 **-** a text box containing the second line of the caption of the button. If this box is empty, the button will be described by means of one line only situated in the middle.

Button-Font **-** a text box provided to specify the font of the button caption characters. The default font is Dialog. Clicking the right mouse button on this box causes activation of the font selection.

Button-normal - color boxes provided to set the color of the caption for the button being normal. The left box is used to determine the text color, middle one – to determine the text shadow color (if the ***Button-shadowing*** option has been activated), and the right one – to set the background color of the caption of the button.

Button-flat - color boxes provided to set the color of the caption for the button being flat. The left box is used to determine the text color, the middle one – to determine the text shadow color (if the ***Button-shadowing*** option has been activated), and the right one – to set the background color of the caption of the button.

Button-pressed - color boxes provided to set the color of the caption for the button being pressed. The left box is used to determine the text color, middle one – to determine the text shadow color (if the ***Button-shadowing*** option has been activated), and the right one – to set the background color of the caption of the button.

Button-Shadow + **-** color boxes provided to specify the illumination feature (left box) and the color of the shadow of the released (normal) button (right box). Illumination is related to the button upper and left sides whilst the shadow is related to the lower right sides. After

selection is made, the window of color selection is displayed.

Button-Shadow- **-** color boxes provided to specify the illumination feature (left box) and the color of the shadow of the pressed button (right box). Illumination is related to the button upper and left sides whilst the shadow is related to the lower and right sides. After selection is made, the window of color selection is displayed.

Button-Frame **-** a check box and a color box provided to specify whether the button should be provided with a frame and to specify the frame color.

Button-Select Protection **-** a check box provided to enable the given button to be selection-protected what means that the execution of an action will require additional pressing on either the *Ctrl* key or *Enter* key.

Button-3-dimensional **-** a check box, which permits drawing of so called three-dimensional buttons.

Button-Transparent **-** a check box provided to draw transparent buttons. In this case, the proper buttons should be created by means of other objects.

Button-Shadowing **-** a check box allowing shadowing the texts on the buttons to be made.

Button-Rounding **-** a check box, which permits rounding the button corners.

Button-Surrounding **-** a numerical box provided to specify the surrounding of the button in terms of pixels (Default = 0). The surrounding of button permits to enlarge the sensitive zone on selection (essential for small buttons).

Password **-** a password combobox, which protects the sending of the controlled variable (four password levels available or without password by default).

Send Immediate - a radio button provided to declare immediate execution of the action after the object is selected, by clicking the mouse button on the area of the button while being under the diagram refreshing mode

Send with Acknowledge - a radio button provided to declare execution of the action after sending acknowledgement only. Acknowledgement can be made by either clicking the button *Grey* + or executing appropriate actions (having selected the button).

Picture-normal **-** a text box provided to specify the name of a bitmap displayed in the area of the button being released (normal). The bitmap name can be either written in the dialog box or selected by clicking the right mouse button. After selection is made, a window will appear for previewing and selecting the bitmaps belonging to the **asix** system bitmap pool.

Picture-flat **-** a text box provided to specify the name of a bitmap displayed in the area of the button being flat. The bitmap name can be either

written in the dialog box or selected by clicking the right mouse button. After selection is made, a window will appear for previewing and selecting the bitmaps belonging to the **asix** system bitmap pool.

Picture-pressed **-** a text box provided to specify the name of a bitmap displayed in the area of the button being pressed. The bitmap name can be either written in the dialog box or selected by clicking the right mouse button. After selection is made, a window will appear for previewing and selecting the bitmaps belonging to the **asix** system bitmap pool.

Justify-to left - justifying a text to the left (within the BUTTON object).

Justify-center - justifying a text to the centre (within the BUTTON object).

Justify-to right - justifying a text to the right (within the BUTTON object).

Get a Key **-** the button, after pressing of which it is necessary to specify the key code (by pressing the appropriate keyboard shortcut) to cause execution of the *BUTTON* object during diagram refreshing. This is an alternative but considerably simpler method to that using a hexadecimal coding (in the ***Code*** box)..

Description **-** a text box provided to display the code (e.g. ***Ctrl-y***) of the key which pressing causes execution of the action under diagram refreshing mode (similarly as with a normal pressing of the button).

Code **-** a text box provided to display and specify a hexadecimal code (e.g. **0x1519**) of the key which pressing causes execution of the action under diagram refreshing mode (similarly as with a normal pressing of the button).

18.4.4. CALCULATOR Object

Dimensions
Object Parameters
Object Name
Times
Limits
Controlled Variable
Calculation
Status Variable Format

The CALCULATOR object is designed for evaluation of ASTEL language expressions. This object remains invisible during refreshing the diagram. CALCULATOR evaluates ASTEL language expression and writes result to a specified system variable. Generally, CALCULATOR should make calculations for other objects placed **on the same** diagram. In the example described evaluation is made in cyclically. This object can also made calculations on

operator's request (by clicking the button with the CALCULATOR type button superimposed and set as „On request").

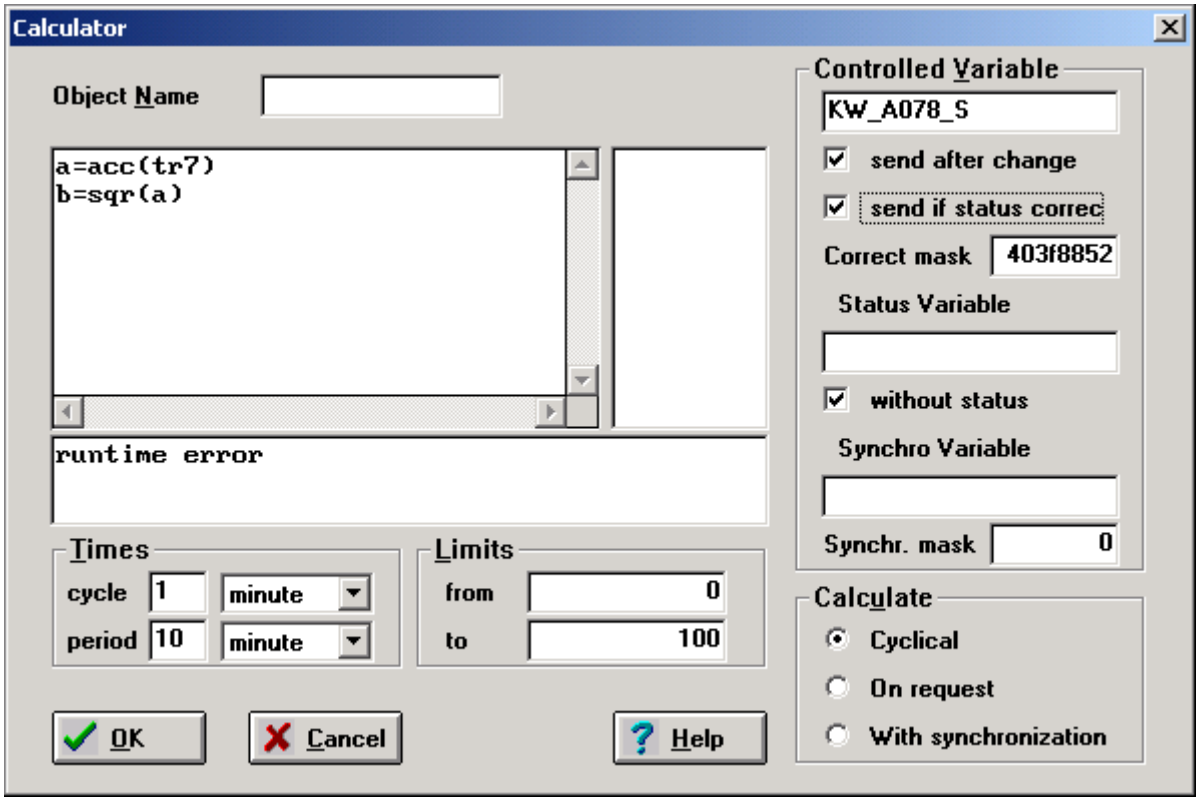


Figure 18-8. The CALCULATOR Object Parameterization Window.

Dimensions

The dimensions of the CALCULATOR object are of no concern (the object is invisible). Under Edit mode, it appears on the screen as a frame (similarly as transparent button-type objects).

Parameters

Object Name - this text box is used to put into it the optional name of the object. If the name of the object will not be defined, its type together with coordinates will appear on the list of objects.

Calculation-Cyclical - a radio button provided to force cyclical evaluation of an expression by the object.

Calculation-On request - a radio button provided to force expression evaluation on operator's request (by clicking on the object box).

Calculation-With Synchronization - a radio button to force evaluation in the case, when the ***Synchro Variable*** bits are set of by the ***Mask*** box.

Controlled Variable - a text box provided to specify a name of the given resulting variable (where the evaluation result will be written in). Specifying this name

is obligatory. With the right mouse button in this box it is possible to open a window containing the list of the available variables to make selection.

Send After Change **-** a check box to insert the result to the controlled variable only when this result is modified (in respect to the last evaluation).

Send if correct - sends the results of declared calculations to ***Result data*** depended on their status. Status is compared with the defined one. If the status is equal to the defined one, the result is rewritten to ***Result data***.

Correctness mask. - declaration of the correctness mask, which is compared with the status of calculations. If the calculation status and ***Correctness mask*** are equal then result can be written to ***Result data***.

Status Variable - text field, which allows specifying the status data name(that the evaluation status will be written to). Specifying this name is not necessary, if ***without status*** option has been chosen. By pressing the right mouse button in this field you can open the window with the list of available variables and make the choice in it. It is possible to specify the name of status data, using notation with **#** character. If in the field of variable name the #_ characters with suffix are entered, then during the application's run it will be treated as the variable with name containing the name of monitored variable instead of the # character.

Without Status **-** a check box to set object parameters in such manner that the status variable is not necessary.

Synchro Variable - text field, which allows specifying the name of synchronizing data (necessary for running in synchronization mode). By pressing the right mouse button in this field you can open the window with the list of available variables and make the choice in it. It is possible to specify the name of status data, using notation with **#** character. If in the field of variable name the #_ characters with suffix are entered, then during the application's run it will be treated as the variable with name containing the name of monitored variable instead of the # character.

Mask**.** **-** a numerical box to insert a hexadecimal mask, the non-zero product of which with the synchronizing variable causes starting of evaluation of the expression.

Times-cycle **-** a numerical box and a combo box provided to determine the refresh cycle of the object by specifying the number of time slices and the unit. This box is important for the cyclical mode of operation.

Times-period **-** a numerical box and a combo box provided to specify the default time period (in the archive)

to which the expressions evaluated by the object are referred. If either no archive variable references are contained in the expressions or the expressions do not use the default period, then the contents of these boxes are of no concern.

Limits-from **-** a numerical box provided to specify the lower value limit for the resulting variable (Default = 0). If the evaluation result contains a lower value then such value is substituted with that lower value.

Limits-to **-** a numerical box providedto specify the upper value limit for the resulting variable (Default = 100). If the evaluation result contains a greater value then such value is substituted with that greater value.

Non described boxes - a text box provided to insert an ASTEL language program (main box below of the object name) and to display the evaluation result (right-hand box) and to display the diagnostic messages as well (the box below). Many successive lines containing expressions may be inserted in the main box. Clicking the right button in this box causes opening of the `***Expression Generator'*** window to permit for expression generation to be made by an user which does not know the language syntax.

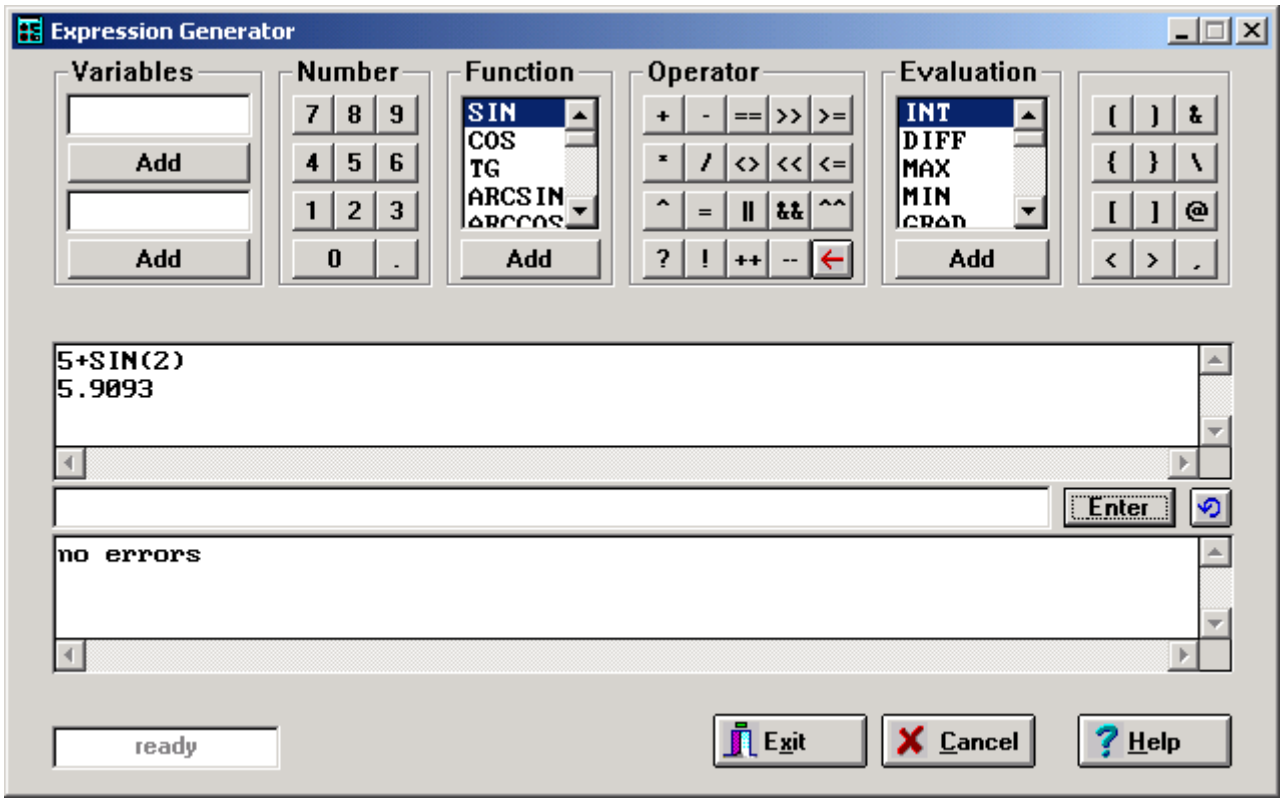


Figure 18-9. 'Expression Generator' Window.

In the case of synchronizing calculation, the evaluation should be forced by setting appropriate parameters of the synchro variable. In the case of cyclical evaluation, the cycle duration time may be determined. The useful feature of the object is the possibility of insertion of a new value to the variable (and the possibility of optional sending of it to the controller) only when the variable changes (the „Send after change" box). The calculated values may be limited within a range determined by the lower and upper limits. Additionally, it is to specify the period of time for which the ASTEL language evaluators are to be evaluated (for example, a 10 minute period means that the evaluator of the average will calculate an average value of the given variable for the last 10 minute period before evaluation starting). If either an expression sequence does not contain evaluators or the evaluators contain the period specified, then the period specified during parameter setting is of no concern.

The ASTEL language expression should be inserted in the expression box placed below the object name. After insertion, each expression may be verified by clicking on the diagnostic box below the expression box. If the expression is erroneous then an error message is displayed in the diagnostic box. If a syntax error is made in the error line, a question mark (?) appears. Clicking on the result box situated on the right-hand side of this box causes evaluation of successive expressions and insertion of the results in the result box. If error occurs, a message is displayed in the diagnostic box.

The expression to be evaluated should be chosen in manner that its evaluation time is up to a dozen of seconds otherwise efficiency of the system, which is loaded with a great amount of time-consuming CALCULATOR objects drops. Such effect of low efficiency (overloading i.e. the situations where the CALCULATOR is reactivated, although the previous evaluation was not finished) may be minimized by assigning relatively long evaluation cycles for the objects, which require long evaluation times. Overloading causes message transmission to the control panel. Such messages can be disabled by inserting in the *.xml application file the following parameters (with use of Architect module):

Architect > *Fields and Computers* > *Miscellaneous* module > *Calculator* tab > ***Overloads*** parameter

The evaluation result status can be written to the status variable and taken by the controller. The variable status is a long fixed-point number. Its format is presented below (the smallest bit number is 0).

Status Variable Format

bit 0 - invalid floating point operation
bit 1 - denormalization of mantissa
bit 2 - division by zero
bit 3 - overflow
bit 4 - underflow
bit 5 - loss of precision
bit 6 - time reverse (error reported by ASPAD)
bit 7 - erroneous argument of rounding
bit 8 - root of a negative number
bit 9 - logarithm of a non positive number
bit 10 - arcus sin cos function domain error
bit 11 - population too small
bit 12 - number conversion error (float-int)

bit 13 - error of channel opening (error reported by ASPAD)
bit 14 - error of function of file positioning (seek)(error reported by ASPAD)
bit 15 - reference to the future (error reported by ASPAD)
bit 16 - lack of variables (error reported by ASPAD)
bit 17 - there are existed later variables (error reported by ASPAD)
bit 18 - hole in archive (error reported by ASPAD)
bit 19 - lack of point in given range (error reported by ASPAD)
bit 20 - variable non refreshed in due time (error reported by ASMEN)
bit 21 - unreliable variable (error reported by ASMEN)
bit 22 - undefined variable (error reported by ASMEN)
bit 23 - transmission error - invalid variable (error reported by ASMEN)
bit 24 - lack of data from server (error reported by ASPAD)
bit 25 - conversion error of text alias
bit 26 - lack of text alias
bit 27 - erroneous format (only for ASTER language operations)
bit 28 - undefined variable of ASTEL language
bit 29 - syntax error of expression
bit 30 - simulated data (for designer's own work only)
bit 31 - erroneous argument of ASTER language evaluator

18.4.5. DATE+TIME Object

Dimensions
Object Parameters
- Object Name
- Attributes
- Mode
- Parameters
- Justify
- Date Format
- Date Separator

The DATE + TIME object enables to display date or time and, alternatively, the date and time together, on the diagram. It is a dynamic object however, which form doesn't depend on system variables, but on the time only. The way of presentation of it can be defined by assigning to it the appropriate font or colors.

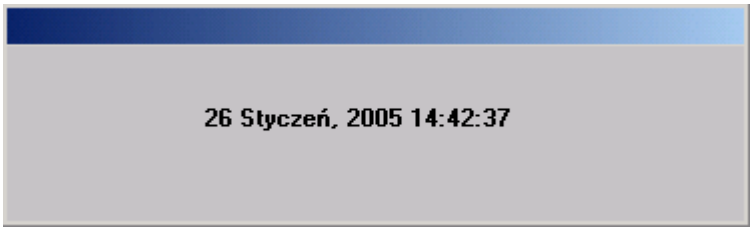


Figure 18-10. The DATE+TIME Example.

Dimensions

The height is set automatically based of the chosen font and no modifications are allowed.

The width is assigned automatically based no the longest possible string (for the date it is October) and modification by use of the mouse is allowed.

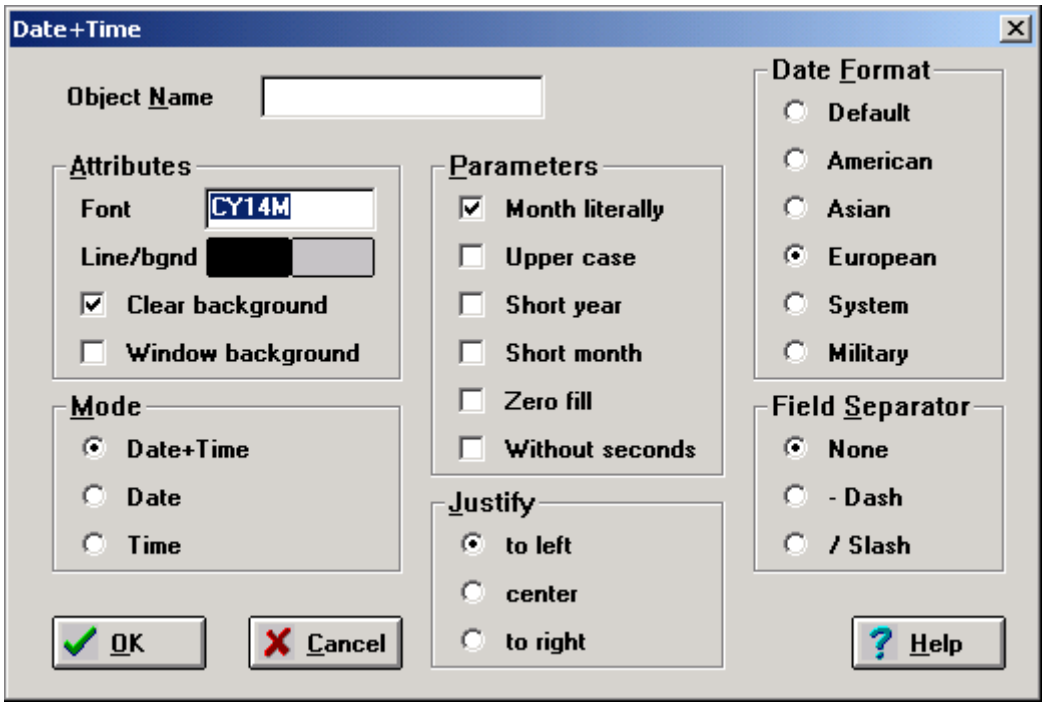


Figure 18-11. The DATE + TIME Object Parameterization Window.

Parameters

Object Name - this text box is used to put into it the optional name of the object. This name appears on the list of objects. If the name of the object will not be defined, its type together with coordinates will appear on the list of objects.

Attributes - Font - a text box that enables to enter the name of the font that is going to be used to display the date and time. If the font name is omitted, the default font (Dialogue) is assigned. Clicking the right mouse button causes the font selection window to be displayed.

Attributes - Line - a color box that enables setting the font color after the color selection window would have been opened.

Attributes - Background - a color box that enables to set the background color after the color selection window would have been opened.

Attributes - Clear Background - a check box that enables clearing the background under the area of the object.

Attributes - Window background - a check box that enables to force drawing of the window background in the background color that has been set for the diagram.

Mode - Date + Time - a radio button that forces both date and time to be displayed.

Mode - Date - a radio button that forces displaying date only.

Mode - Time - a radio button that forces displaying time only.

Parameters - Month literally - a check box that forces displaying names of months in words, e.g. "25th November 98"

Parameters - Upper Case - a check box that forces displaying names of months in capital letters (upper case), e.g. "3rd MAY 1996".

Parameters - Short year - a check box that forces displaying the year in short form, e.g. "7 10 **93**‘

Parameters - Short month - a check box that forces displaying the month in the form abbreviated to three letters, e.g. "7 **Oct.** 93‘

Parameters - Zero fill - a check box that forces displaying the month (and year if allowed) numbers preceded by leading zeros, e.g. "07/01/93‘

Parameters - Without seconds - a check box that forces displaying the time without the second box, e.g. "12:45"

Justify - To the Left - a radio button that forces displaying the text in the object box justified to left.

Justify - Center - a radio button that forces centering of the text in the object box.

Justify - To the Right - a radio button that forces displaying the text in the object box justified to right.

Date Format - Default - a check box that forces displaying the date according to system settings.

Date Format – American - a check box that forces displaying the date in American format, e.g. **"March 28, 1990"**.

Date Format - Asian - a check box that forces displaying the date in Asian format, e.g. **"1990 March 28"**.

Date Format - European - a check box that forces displaying the date in European format, e.g. **"28th March 1990"**.

Date Format - System - a check box that forces use of system date format, e.g. **"3/28/90"**.

Date Format - Military - a check box that forces displaying the date in military format, e.g. **"28 Mar 90"**."

Date Separator - None - a check box that forces displaying the date with no separator, e.g. "**3 28 90"**.

Date Separator - Dash - a check box that forces use the hyphen as a date separator e.g. "**3-28-90"**.

Date Separator - Slash - a check box that forces use the slash as a date separator e.g. "**3/28/90"**.

By default, the following options are set: ***Mode - Date + Time, Parameters - Month in Words, Justify - To Left, Date Format - European*** and ***Date Separator - None***.

It is recommended to use the two independent objects instead of the object *DATE + TIME* (then the date will not be refreshed every second but only every 24 hours). Moreover, use of the elite character is preferable (with fixed width of the character) - you avoid the effect of dynamic change of the width of the object.

18.4.6. ELLIPSE Object

Dimensions
Object Parameters
- Object Name
- Line Color
- Fill Color
- Initial Angle
- Final Angle
- Line Thickness
- Fill Inside
- Window Background
- Draw Circle

The ELIPSE object enables displaying the ellipse or its sector on the diagram. It is the static object. Definition of dimensions and colors as well as the initial and final angle are to be defined. Drawing a circle, as a special case of ellipse is also possible.

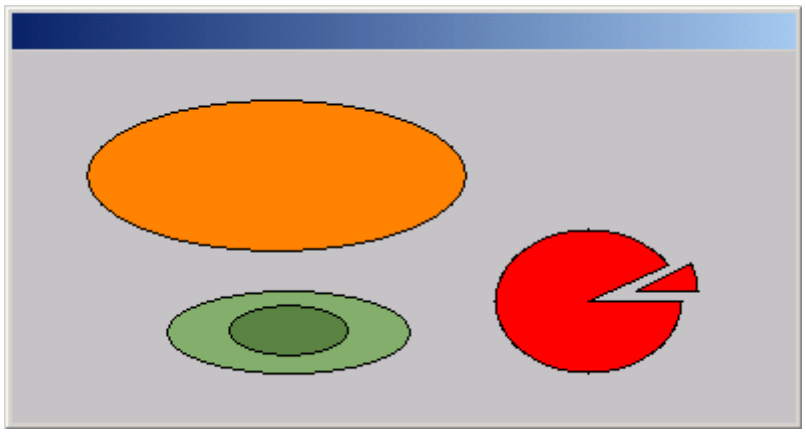


Figure 18-12. The ELIPSE Example.

Dimensions

The height and width are set with use of the mouse.

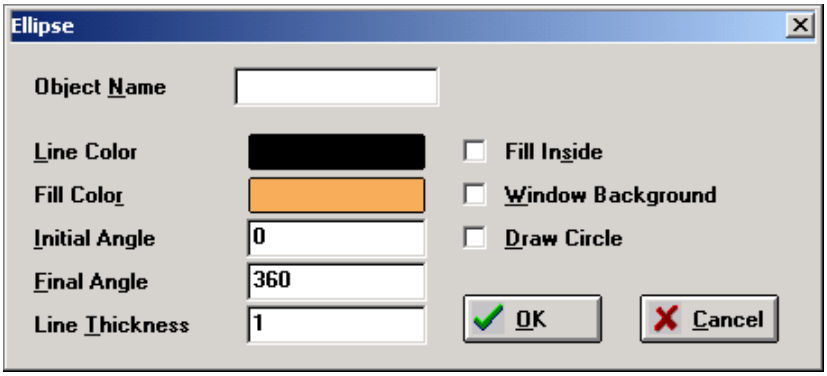


Figure 18-13. The ELIPSE Object Parameterization Window.

Parameters

Object Name - this text box is used to put into it the optional name of the object. This name appears on the list of objects. If the name of the object will not be defined, its type together with coordinates will appear on the list of objects.

Line Color - a color box that enables setting the bordering line color after the color selection window would have been opened.

Fill Color -a color box that enables setting the internal area color after the color selection window would have been opened.

Initial Angle - a numerical box that enables setting the initial angle of the elliptic arc (0 by default).

Final Angle - a numerical box that enables setting the final angle of the elliptic arc (360 by default).

Line Thickness - a numerical box that enables setting the width of the line (in pixels) of the ellipse.

Fill Inside - a check box that enables filling the internal area of the ellipse.

Window Background - a check box that enables drawing the internal area of the ellipse in the background color that has been set for the diagram.

Draw Circle - a check box that enables of drawing a circle

The ellipse (or its sector) is drawn with the line of the defined width. The initial angle should be less than the final one and their difference should be less or equal 360 grades. The inner area can be left unfilled if the single line has been selected only.

18.4.7. ELLIPSES Object

Dimensions
Object Parameters
- Object Name
- Parameters
- Send
- State
- Bit Mask
- Line
- Background
- Blink.
- Coding
- Variable
- Angles

Examplary Coding

The ELIPSES object is a variation of the ELLIPSE one and enables changing colors of the ellipse that has been drawn depending on status of the monitored variable. It is the dynamic object (whether it has more than one state). Optionally, the ELLIPSES object can be also the selectable object and may be used for sending of the defined value to the control variable. Sending the value can be protected by password.

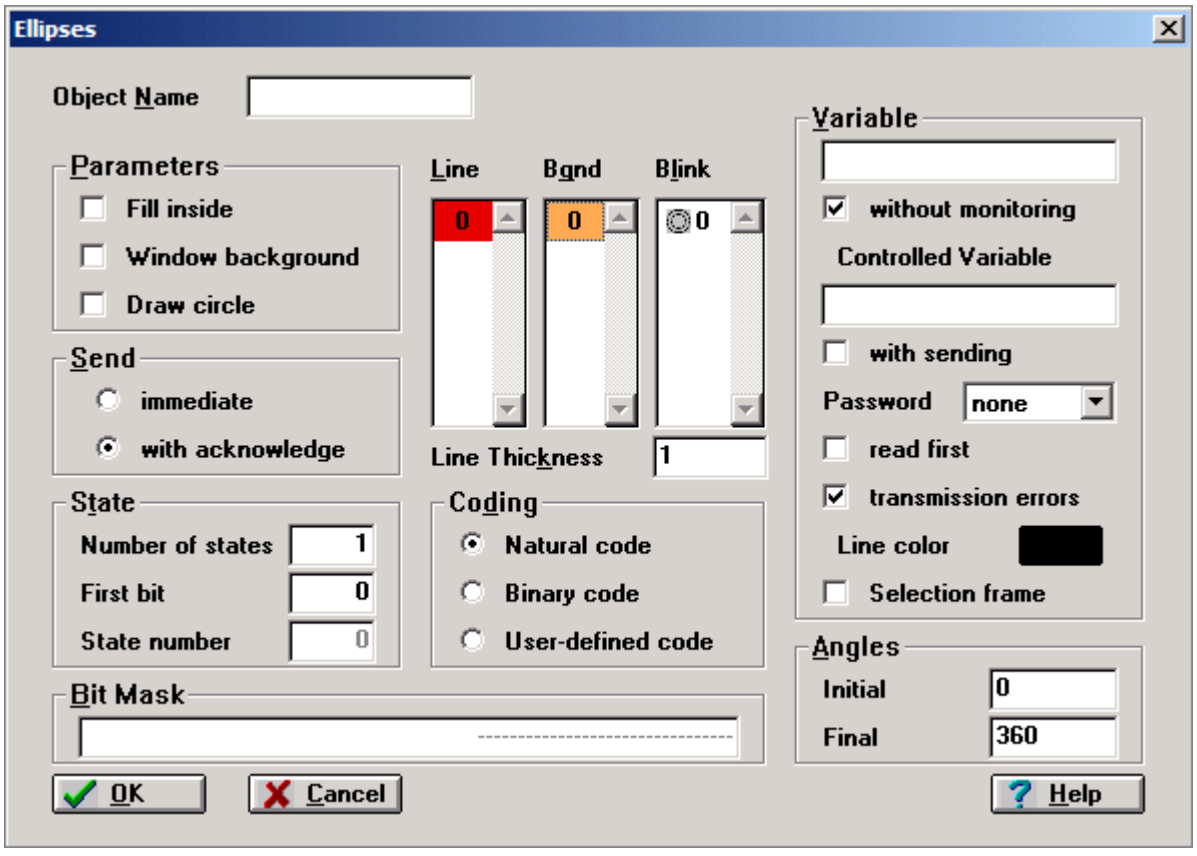


Figure 18-14. The ELIPSES Object Parameterization Window.

Dimensions

Dimensions of the ellipse are defined with use of the mouse.

By default, the object has only one state and it is then the static object. Communication errors are alerted by change of color.

Parameters

Object Name - this text box is used to put into it the optional name of the object. This name appears on the list of objects. If the name of the object will not be defined, its type together with coordinates will appear on the list of objects.

Parameters - Fill Inside - a check box that enables filling of internal ellipse area with the declared color.

Parameters - Window Background **-** a check box that enables filling of internal ellipse area in the background color that has been set for the window independently on the color set internal color.

Parameters - Draw Circle **-** a check box that enables forcing of drawing a circle

Send - Immediate - a radio button that enables declaration of immediate sending of the value after having selected of the corresponding ellipse by the mouse clicking in the object box in the diagram refreshment (the above is possible only whether the parameter ***with sending*** of the object has been set). This type of transmission is reasonable whether the object has two states, otherwise checking of the object status executed as a result of scanning the consecutive states will cause sending of the series of values. The value by object is the number that corresponds to the state number of object.

Send - With acknowledge - a radio button that enables declaration of sending the value after having selected of the corresponding ellipse by means of the (multiple) mouse clicking in the object box in the diagram refreshment mode (the above is possible only whether the parameter ***with sending*** of the object has been set) and further acknowledging. Acknowledge can be carried out by pressing the *Grey+* key or by executing the appropriate action. This type of transmission is reasonable whether the object has more states. The value by object is the number that corresponds to the state number of object.

State - Number of States - a numeric box that enables declaring a number of states of the object. By default the object has 1 state and then it is the static object. Maximum number of states is 33 for natural coding, 64 for binary coding and user-defined coding. The individual states

correspond with values that are result of the chosen coding mode.

State - First bit - a numeric box that enables declaration of the first (LSB) bit of the monitored data, value of that is displayed as one of many declared ellipses. The content of that box (significant only for natural and binary coding mode) enables unambiguous definition of location in the word of the bit group that controls the displayed ellipses. The first bit determines the less significant bit (LSB) and the size of the group defines the coding mode.

State - State Number - a numeric box that enables displaying the selected state number only. Selection of state is carried out by clicking on the appropriate item of one of the two colors list.

Line - a color list that are referred to borders of individual ellipses. Change of color can be carried out by having selected the appropriate color and striking the *Enter* key. The color selection window will then appear. The attempt to declare color for the non-existing state will be unsuccessful. Numbers of states (from 0 to n) are displayed on the color list in black.

Background - a color list that are referred to internal ellipses area. Selection of the appropriate color is to be made in the same way as a line color selection (see above).

Blink - opening the blinking attribute window.

Angles - Initial **-** a numerical box that enables setting the initial angle of the elliptic arc (0 by default).

Angles - Final **-** a numerical box that enables setting the final angle of the elliptic arc (360 by default).

Coding - Natural code - a radio button that enables selection of the natural coding mode (coding modes are going to be discussed after the list of parameters).

Coding - Binary code - a radio button that enables selection of the binary coding mode (coding modes are going to be discussed after the list of parameters).

Coding - User-defined code - a radio button that enables selection of the user-defined coding mode (coding modes are going to be discussed after the list of parameters).

Variable - a text box that enables declaration of the name of the monitored data. Declaration of this name is not necessary whether the option ***without monitoring*** has been selected. Clicking the right mouse button on this box you can open the window with the list of available variables and make selection within this window.

Controlled variable - a text box that enables declaration of the controlled data name. Declaration of this name is obligatory whether the option ***with sending*** has been selected. Clicking the right mouse button on this box you can open the window

with the list of available variables and make selection within this window.

Without monitoring **-** a check box that enables setting parameters of the object in such a manner, that the displayed status is not dependent on any system variable and will serve only for indication of the value that is to be sent.

With sending **-** a check box that enables setting parameters in such a manner, that the value corresponding with the state of the object would be sent to the system variable. Sending of value is executed after having the object selected e.g. by mouse click, the further process depends on the mode of sending.

Password - a combo list box of the password that protects sending of the controls (four levels of password can be selected, no password is the default setting).

Read First **-** a check box that enables forcing of reading the value before controls will be sent. In this case the logic sum of the sent value and value of the given variable that has been read from the controller are written to the controlled variable, while the group of bits being set by the object is set to 0 (zero).

Transmission errors **-** a check box that enables alerting of communication errors (the object is to be struck out). The option is set by default.

Line Color - a color box of the line that strikes out the object if transmission error has been occurred. Selection of this check box will cause opening of the color selection window.

Bit Mask - a numeric box that enables displaying the monitored variable (and simultaneously the variable that can be sent out from the object, whether the parameter ***with sending*** has been set) that correspond with the given state. In user-defined coding mode the diagram can be set by insertion of the hexadecimal or binary code corresponding with the given state, while *don't care* bits, if needed, are marked, by the "-" symbol.

The ***'Coder'*** window is used to defie the states in user-defined coding mode:

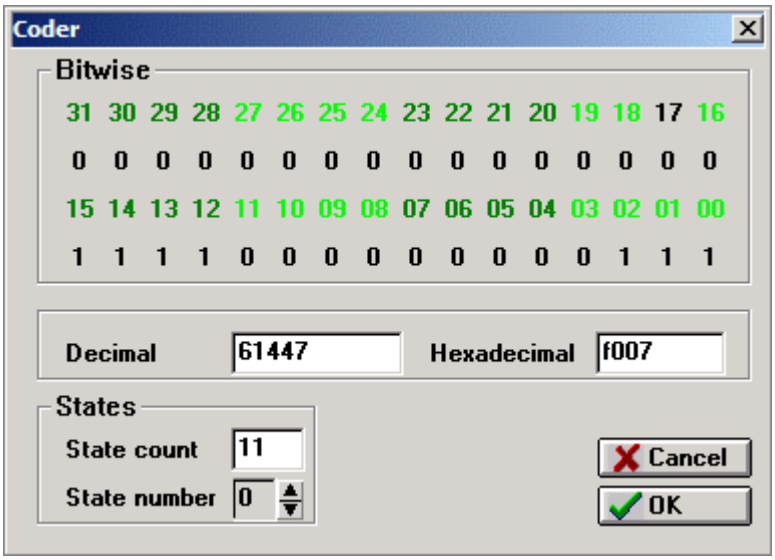


Figure 18-15. The 'Coder' Window.

The number of states recognized by the object is passed in *State count* field. This value may be also set in the main window for object defining. The state the code of which should be defined is selected in *State number* field. While setting the number in *Bitwise* frame, the current state coding is displayed - to change it, one should click with mouse on a bit value. Successive click operations cause change of the value in the following sequence:

- – bit value is unimportant
1 – bit has to be set
0 – bit has to be set to zero

Sate coding is automatically stored in the moment of choosing the other state.

The *Decimal* and *Hexadecimal* fields display decimal and hexadecimal value of defined state.

Selection frame - a check box that enables displaying an additional so called *Selection frame* (border of the object) after the object has been selected while diagram refreshing. This frame allows to guess easy, which object has been selected as the last one.

The object, while refreshing, can be selected by means of the mouse cursor or the *Tab* key (*Shift + Tab).* After having selected the object, the desired state of it (indicated by colors of the border and internal area) can be selected by means of mouse cursor or the *Enter* key. Sending of the state is executed automatically after selection or acknowledges. Striking of the "Grey+" key or executing the appropriate action can carry out acknowledge. You can cancel selection striking the *Esc* key or clicking the right mouse button.

Usage of the two conversion functions is possible: values of the first one are equal to indexes of individual colors, and the second conversion function converts number of the object state to the value being sent (the above is

referred to the option *with sending*). Decoding of the fixed-point monitored value (and coding of the sent values, if necessary) is also possible by means of one of the three strategies built-in into the object.

Natural Coding

The group of bits is checked. Number of bits is equal to the *number of states*, but number of the LSB defines "offset". The group of bits can consist only one value of "1". Number of the state is calculated as position of the "1" within the group of bits (the LSB position is number one). The zero state corresponds with the group of all zeros.

Binary Coding

The group of bits is checked. Number of bits is equal to the binary logarithm of *number of states*, but number of the LSB defines "offset". Number of state is calculated as content of the tested bit group.

User-Defined Coding

The whole word is checked. The "offset" parameter is ignored. The user assigns a string of ones and zeros (the *don't carry* states can be masked by "-") to each state. Number of state is calculated as the first of states, that the user assigned the same bits on the appropriate positions (regardless to the masked bits).

For the object, when parameter *with sending* has been set for it, the use of option *read first* is possible. In such a case the logical sum of the value being sent and the given value that has been read from the controller (the bits being controlled by the object are reset to zero) is written to the variable.

EXAMPLE

The object has four states and binary coding has been chosen. For encoding of four binary states (00, 01, 10, 11) the group of two bits is necessary in the word. The first (LSB) bit of the group determines content of the box named ***State - First bit***. If this content is equal e.g. 4, that state only of the fifth and sixth bit influents on state of the object. The other bits are not referred to displaying the object what is being illustrated below.

```
----------xx----
```

("-" denotes 'don't carry' bit, but 'x' denotes the bit that influents on state of the object)

If you define the four ellipses that correspond with the four states mentioned above, e.g. yellow, green, blue and red ones (in this order), the yellow ellipse correspond with the "00" state, the green one with the "01" state, etc. In case of writing, whether the red ellipse is selected, the state corresponding with it, i.e. "11" on the 5^{th} and 6^{th} bits i.e. 48 in decimal, are sent to the variable. Whether the box ***read first*** is ticked the described procedure operates slightly different - the given variable is previously read, then its 5^{th} and 6^{th} bits are reset to zero, next the bits that correspond with the color of the selected ellipse are put in on the place of those bits.

```
1000000000100000    / the variable that has been read
1000000000--0000    / the variable that has been read with "-" bits reset
to 0
----------11----    / the bits set by the object
1000000000110000    / the variable that is to be sent
```

18.4.8. EXPESSION Object

Dimensions
Object Parameters
- Object Name
- Inserting the ASTEL Code
- Color
- Ranges
- Calculate
- Presentation
- Justify
- Position
- Times

Object Operation
Description of Formats
- Examples

The EXPRESSION object permits for evaluation of ASTEL language expressions and displays a numerical result of the last evaluation too. Thus, it is a combination of the CALCULATOR and NUMBER objects. Evaluation may be carried out cyclically. This object can also made calculations on operator's request (after selecting the object).

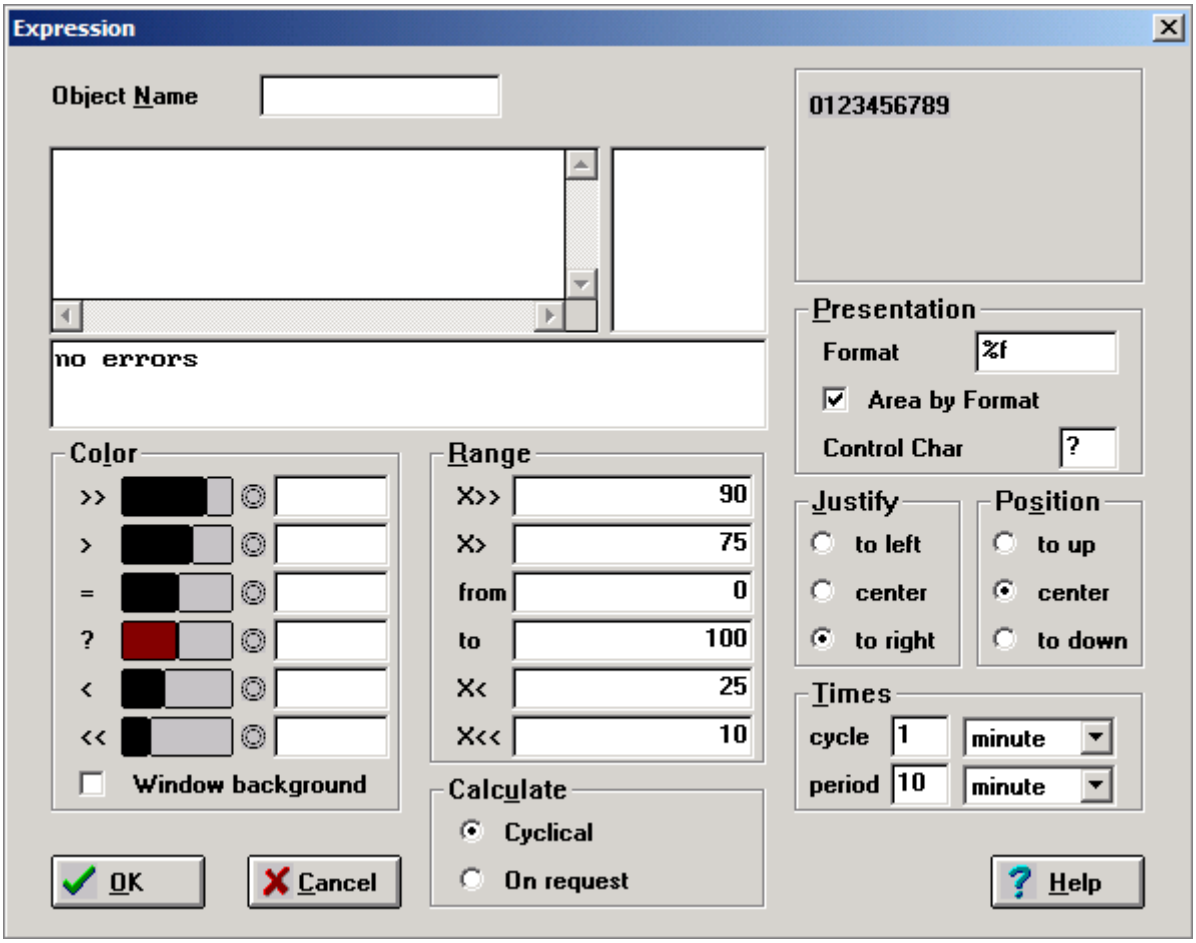


Figure 18-16. The EXPRESSION Object Parameterization Window.

Dimensions

Both the height and the width is specified automatically according to the specified format for the possibly longest string resulting from the given range. The size of the object can be enlarged using the mouse. By specifying the Area by Format attribute, the size is changed automatically to the size as resulting from the format.

Parameters

Object Name - this text box is used for to put in the optional name of the object. If the name of the object will not be defined, its type together with coordinates will appear on the list of objects.

Non described boxes **-** a text box to insert an ASTEL language program (main box below of the object name) and to display the evaluation result (right-hand box) and to display the diagnostic messages as well (the box underneath). Many successive lines containing expressions may be inserted in the main box. Clicking the right button in this

box causes opening of the ***Expression Generator*** window (see description of the *CALCULATOR* object) to permit for expression generation to be made by a user which does not know the language syntax.

Color ->> - color boxes to specify the color of the numeric characters (left box) and the background (right box) for situation of exceeding of the upper alarm limit.

Color -> - color boxes to specify the color of the numeric characters (left box) and the background (right box) for situation of exceeding of the upper warning limit.

Color -= - color boxes to specify the basic color of the numeric characters (left box) and the background (right box) for the displayed number.

Color -? - color boxes to specify the color of the numeric characters (left box and middle box) and the background (right box), of the displayed number for situation of transmission error occurrence (left box) and of measurement error occurrence (middle box).

Color -< - color boxes to specify the color of the numeric characters (left box) and the background (right box) for situation of exceeding of the lower warning limit.

Color -<< - color boxes to specify the color of the numeric characters (left box) and the background (right box) for situation of exceeding of the lower alarm limit.

Color-Window background - a check box to force the background color being the same as the ***Background*** window color.

Color - to the right from the fields of color selection for particular limits, the blinking attribute indicators are located. Further to the right is a column of font selection fields. The fonts may be selected individually for each of the limits, which allows e.g. declaring bigger font for alarm situations, to make it easier for the Operator, to notice exceeding technological limits.

Range -X>> - a numerical-text box to specify either a numerical value of the upper alarm limit or a name of the variable which contains this limit. The default value is 90. When the variable name is to be specified, its logic condition may be additionally specified in the box situated rightward. See the next sections for details on the conditions.

Range -X> - a numerical-text box to specify either a numerical value of the upper warning limit or a name of the variable, which contains this limit. The default value is 75. When the variable name is to be specified, its logic condition may be additionally specified in the box situated

rightward. See the next sections for details on the conditions.

Range - from - a numerical-text box to specify a numerical value of the lower range. The default value is 0. The lower numbers will be displayed. The value is used only to determine the area occupied by the object.

Range - to - a numerical-text box to specify a numerical value of the upper range. The default value is 100. The greater numbers will be displayed. The value is used only to determine the area occupied by the object.

Range -X< - a numerical-text box to specify either a numerical value of the lower warning limit or a name of the variable which contains this limit. The default value is 25. When the variable name is to be specified, its logic condition may be additionally specified in the box situated rightward. See the next sections for details on the conditions.

Range -X<< - a numerical-text box to specify either a numerical value of the lower alarm limit or a name of the variable which contains this limit. The default value is 10. When the variable name is to be specified, its logic condition may be additionally specified in the box situated rightward. See the next sections for details on the conditions.

Calculate-Cyclical - a radio button to force cyclical evaluation of an expression by the object.

Calculate-On request - a radio button to force evaluation on operator's request (by clicking on the object box).

Presentation-Font - a text box to specify the font for the number to be displayed. The default font is Dialog. Clicking the right mouse button on this box causes activation of the font selection window.

Presentation- Format - a text box to declare the format for the number to be displayed. The allowable formats are derived from the C language standard and are described in the next chapters. The default format is %d (integer).

Presentation- Area by Format **-** a check box to force the area occupied by the object in ***by Format*** accordance with the format and the font. For example, a %5u format means that the area occupied by the object will correspond to 5 numeric characters having dimensions determined by the selected font.

Control character - a text box to declare the control character (Default = ?), which is used for error signalling.

Justify-to left - a radio button to force left alignment of the number within the object area.

Justify-center - a radio button to force centering of the number within the object area.

Justify-to right	- a radio button to force right alignment of the number within the object area.
Position – to up	- selection field forcing vertical justification of the number to the upper edge of the object's area.
Position – center	- selection field forcing vertical centering of the number inside the object's area.
Position – to down	- selection field forcing vertical justification of the number to the lower edge of the object's area.
Times - cycle	**-** a numerical box and a combo box to determine the object refresh cycle by specifying the number of time slices and the unit of measure. Important for a cyclical mode of operation.
Times - period	**-** a numerical box and a combo box to specify the default period (in the archive) being a time reference for the expressions evaluated by the object. If expressions do not contain references to the archived data or they do not use the default period then the contents of these boxes are of no concern.

NOTICE Empty texts and the texts consisted of white spaces are acceptable in limit fields and using them is treated as abandoning verification.

Object Operation

When a calculation error occurs, (for example, trying to calculate a logarithm of a negative number), the number is substituted with a sequence of control characters.

The displayed number may change its color on exceeding limits. Exceeding can be specified as either the numbers (fixed) or the system variable names. If the exceeding is specified as a system variable name, a logic condition to be met can be specified in the box close to the name (rightwards).

The logic condition is a sequence consisting of a „&" or „|" character followed by a hexadecimal number (condition). With the „&" character, condition is met if the logic product of the variable and the condition equals to the condition whilst with the „|" character, condition is met if the logic sum of the variable and the condition is different from zero. In other words, in the first case, to change the attribute it is necessary to have all the bits set in the given variable whilst in the second case at least one bit of the condition must be 1. When specifying the condition, the „|" character may be neglected.

In the case of an on request calculation, the operator may force the calculation by selecting the object. To make a cyclical evaluation it is necessary to determine the cycle duration. Additionally, in both cases it is necessary to specify the period for which the ASTEL language evaluators will be evaluated (for example, specifying a 10 minute period means that the average evaluator will calculate an average value of the given variable for the period of last 10 minutes before commencing the evaluation). If a sequence of expressions does not contain evaluators or they contain the specified period as a parameter, then the period specified during parameter setting is of no concern.

The ASTEL language expression should be inserted in the expression box placed below the object name. After insertion, each expression may be verified by clicking on the diagnostic box below the expression box. If the expression is erroneous then an error message is displayed in the diagnostic box. If a syntax error is made in the error line, a question mark (?) appears. Clicking on the result box situated on the right-hand side of this box causes evaluation of successive expressions and insertion of the results in the result box. If error occurs, a message is displayed in the diagnostic box.

Description of Formats

The format parameter has to be compatible with the following syntax:

[%][[0]width][.precision][date_format][time_format][l|h]type

where:

%	- optional beginning of a format;
type	- formatting method: d,i - decimal signed integer, u - decimal unsigned integer, o - octal number, x - hexadecimal number using small letters abcdef, X - hexadecimal number using capital letters ABCDEF, B - number in BCD code, F,e,g,E,G - floating-point number, D - date, input datum is treated as a number of seconds since 1970-01-01-00:00, 32 bites; T - time, input datum is treated as number of seconds, 32 bites;
[0]width	- minimum field width; adding the 0 before the number of width means that the field will be completed by the 0 figures on the left; the element is used only for d,i,u,o,x,X,B,f,e,g,E,G formats;
precision	- quantity of fractal digits for f,e,g,E,G;
l/h	- additional input data type definition: 1-32 bits, h – 16 bits; only for d,i,u,o,x,X,B formats;
data format	- detailed date format definition used only for D; it consists of sequence of characters defining separators and date components in arbitrary sequence; there are the following components: y-year, m-month, d-day, h –hour, n-minute, s-second; no date format definition causes formating acording to the pattern: y/m/d h:n:s;
time_format	- detailed tme format definition used only for T; it consists of sequence of characters defining separators and time components in arbitrary sequence; there are the following components: d- days, h –hours, n-minutes, s-seconds, z-milliseconds; no time format definition causes formating acording to the pattern: h:n:s;

NOTICE If the field d is used, then the field h are not higher then 23, e.g. if the format "h:n:sT" displays „46:10:00", the format „d h:n:sT" displays „1 22:10:00".

NOTICE You may set the letter "I" before the letter "T"– that means that an input datum is a number of seconds (and not milliseconds).

> **NOTICE** For D and T formats it is possible to set an empty text that corresponds to 0, e.g.:
> 1970-1-1 0:0:0 for D,
> 0:0:0 for T.

The concept of formats is derived from the C programming language.

EXAMPLES

%i	- a fixed-point number, all significant digits are specified.
%7lu	- a long fixed-point number, without sign, seven positions.
%6.2f	- a floating-point number, decimal, six positions, two decimal positions.
h:n:s:zT	- time format with milliseconds.
T	- there is no neeed to put % before T.
y/m/dD	- date format without time.
y-m-dD	- date format of the following form: 2005-11-30.

18.4.9. LINE and LINE HV Objects

Dimensions
Object Parameters
- Object Name
- Line Thickness
- Type
- Line Color
- Break Color
- Break
- Pattern
- Definition
- Round Line Edges

The LINE and LINE HV objects enable displaying the line segment on the diagram. In case of the LINE_HV object this line segment can be either horizontal or vertical. It is the static object. Definition of color and line width is possible.

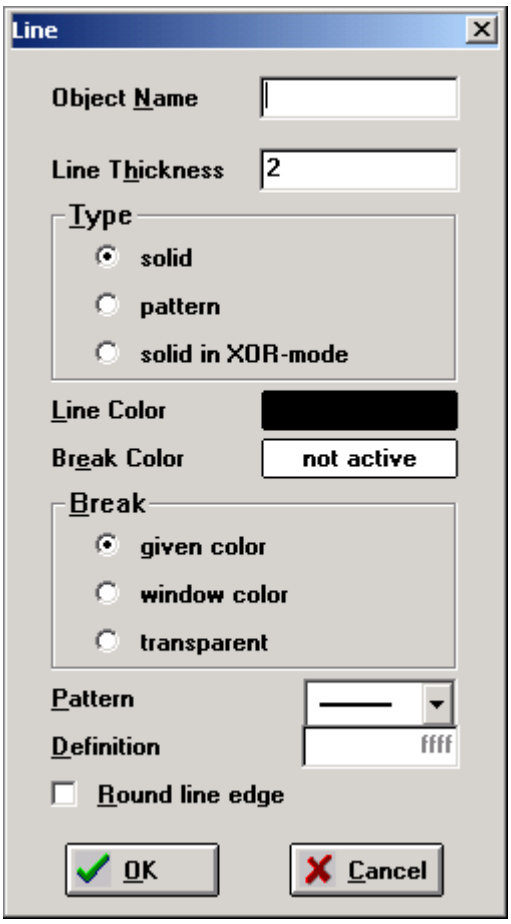


Figure 18-17. The LINE and LINE HV Object Parameterization Window.

Dimensions

Coordinates of the begin and the end of the line segment are defined by means of the mouse.

Parameters

Object Name - this text box is used for to put in the optional name of the object. If the name of the object will not be defined, its type together with coordinates will appear on the list of objects.

Line Thickness **-** a text box that enables defining of the line width in pixels. For dotted (not solid) lines only odd values of line width are allowed.

Type - solid - a radio button that enables selection of the normal (solid) line.

Type - pattern - a radio button that enables selection of the dotted (dashed) line.

Type - solid in XOR mode - a radio button that enables selection of the normal (solid) line being drawn in XOR mode, i.e. the line color is a result of XOR operation of the line itself and background, where the line is drawn.

Line Color	- a color box that enables selection of the line color. After having this box selected the color selection window appears.
Break color	- a color box that enables selection of the gap color (whether the line is dotted). Otherwise the window will be inactive. After having this box selected the color selection window appears.
Break - given color	- a radio button that enables selection of such a mode of drawing of dotted (dashed) line, while the line gaps are drawn in the defined color.
Break - window color	- a radio button that enables selection of such a mode of drawing of dotted (dashed) line, while the line gaps are drawn in the color that has been set for the window.
Break - transparent	- a radio button that enables selection of such a mode of drawing of dotted (dashed) line, while the line gaps would be transparent.
Pattern	- a combo box that enables selection of a pattern for the dotted/dashed line (some patterns are predefined and the user-defined pattern defined in the following box can be selected as well). The solid line is the default pattern.
Definition	- a text box that enables displaying the line pattern or defining the own one. The pattern is displayed as a hexadecimal number bits of it correspond with consecutive points of the line. (E.g. the displayed pattern "ffff" stands for the solid line). This box can be edited while the user-defined pattern has been selected (by use of the option **Pattern**).
Round line edges	- a check box that enables rounding of the line edges. It is allowed only for the dotted /dashed lines.

Colors of the line *drawn in XOR mode* is set as the XOR function of the declared line color and the background color. This is due to provide permanent visibility of the line on the background color and, when passing over the backgrounds of various colors, the line color will be automatically changed.

18.4.10. LINES and LINES HV Objects

Dimensions
Object Parameters
Object Name
Parameters
Sending
State
Line
Blink.
Coding
Variable
Bit mask
Coding Example

The LINES and LINES HV objects enable displaying the lines on the diagram. In case of the LINES HV object this line can be either horizontal or vertical. The displayed line may change its color depending on the state of the monitored value. It is the dynamic object (if it has more than one state). Optionally, the LINES object can be also the selectable object and may be used for sending of the defined value to the controlled variable. Sending of contents can be protected by password. Displaying of the so-called "selection frame" that appears by default after the object has been selected, may be locked.

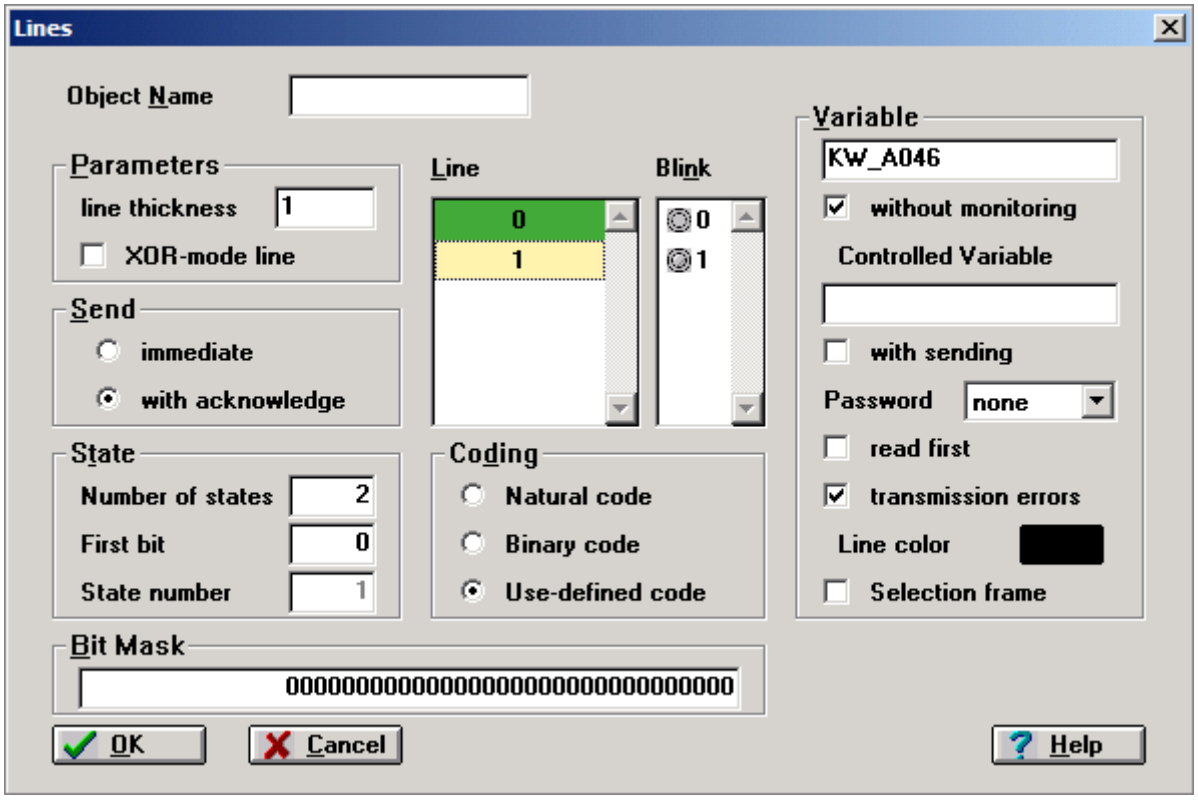


Figure 18-18. The LINES and LINES HV Object Parameterization Window.

Dimensions

Coordinates of the beginning and the end of the line segment are defined by means of the mouse.

Parameters

Object Name - this text box is used for to put in the optional name of the object. If the name of the object will not be defined, its type together with coordinates will appear on the list of objects.

Parameters - Line thickness - a text box that enables defining of the line width.

Parameters XOR-mode line **-** a check box that enables forcing that the line is being drawn in XOR mode, i.e. the line color for such a mode is a result of XOR operation of the declared line color itself and background color. The following will cause, that the line is always visible, and changes its color while passing over various background colors.

Send - Immediate - a radio button that enables declaration of immediate sending of the value after having selected of the corresponding line by means of the mouse click in the object box in the diagram refreshment mode (the above is possible only whether the parameter ***with sending*** of the object has been set). This type of transmission is reasonable whether the object has two states, otherwise checking of the object status executed as a result of scanning the consecutive states will cause sending of the series of values. The value by object is the number that corresponds to the state number of object.

Send - With acknowledge - a radio button that enables declaration of sending of the value after having selected of the corresponding line by means of the (multiple) mouse click in the object box in the diagram refreshment mode (the above is possible only whether the parameter ***with sending*** of the object has been set) and further acknowledge. Acknowledge can be carried out by striking the "*Grey+*" key or by executing the appropriate action. This type of transmission is reasonable whether the object has more states. The value by object is the number that corresponds to the state number of object.

State - Number of States - a numeric box that enables declaring a number of states of the object. By default the object has 1 state and then it is the static object. Maximum number of states is 33 for natural coding, 64 for binary coding and user-defined coding. The individual states

correspond with values that are result of the chosen coding mode.

State - First bit - a numeric box that enables declaration of the first (LSB) bit of the monitored data, value of that is displayed as one of many declared lines. The content of that box (significant only for natural and binary coding mode) enables unambiguous definition of location in the word of the bit group that controls the displayed lines in the word. The first bit determines the less significant bit (LSB) and the size of the group defines the coding mode.

State - State Number - a numeric box that enables displaying the selected state number only. Selection of the state is carried out by clicking on the appropriate item of the colors list.

Color List - colors list that correspond with individual lines. The color can be changed by selecting the appropriate item on the list and pressing the *Enter* key. The color selection window will then appear. The attempt to declare color for the non-existing state will be unsuccessful. Numbers of states (from 0 to n) are displayed on the color list in black.

***Blink*.** - opening the window blinking attribute parameterization.

Coding - Natural code - a radio button that enables selection of the natural coding mode (coding modes are going to be discussed after the list of parameters).

Coding - Binary code - a radio button that enables selection of the binary coding mode (coding modes are going to be discussed after the list of parameters).

Coding - User-defined code - a radio button that enables selection of the user-defined coding mode (coding modes are going to be discussed after the list of parameters).

Variable - a text box that enables declaring the name of the monitored data. Declaration of this name is not necessary whether the option ***without monitoring*** has been selected. Clicking the right mouse button on this box you can open the window with the list of available variables and make selection within this window.

Controlled variable - a text field that allows specifying the name of control variable. Specifying this name is necessary, if ***with control*** option has been selected. By pressing the right mouse button in this field you can open the window with the list of available variables and make the choice in it. It is possible to specify the name of controlled data, using notation with **#** character. If in the field of variable name the #_ characters with suffix are entered, then during the application's run it will be treated as the variable with name containing the name of monitored variable instead of the # character.

Without monitoring **-** a check box that enables setting parameters of the object in such a manner, that the displayed status doesn't depend on any system variable and will be used only to indicate the value that is to be sent.

With sending **-** a check box that enables setting parameters in such a way, that value corresponding to the state of the object will be sent to the system variable. Sending the value is executed after having the object selected, e.g. by mouse clicking, the further process depends on the transfer mode.

Password - a combo box of the password that protects against sending the controls (four levels of password can be selected, no password is the default setting).

Read First **-** a check box that enables forcing of reading the value before controls will be sent. In this case the logic sum of the sent value and value of the given variable that has been read from the controller are written to the controlled variable, while the group of bits being set by the object is set to 0 (zero).

Transmission error **-** a check box that enables alerting of communication errors (the object is to be struck out). The option is set by default.

Line Color - a color box of the line that strikes out the object if transmission error has been occurred. Selection of this check box will cause opening of the color selection window.

Bit Mask - a numeric box that enables displaying the monitored variable (and simultaneously the variable that can be sent out from the object, whether the parameter ***with sending*** has been set) that correspond with the given state. In user-defined coding mode the mask can be set by insertion of the hexadecimal or binary code corresponding with the given state, while *don't care* bits, if needed, are marked, by the "-" symbol.

The ***'Coder'*** window is used to define the states in user-defined coding mode:

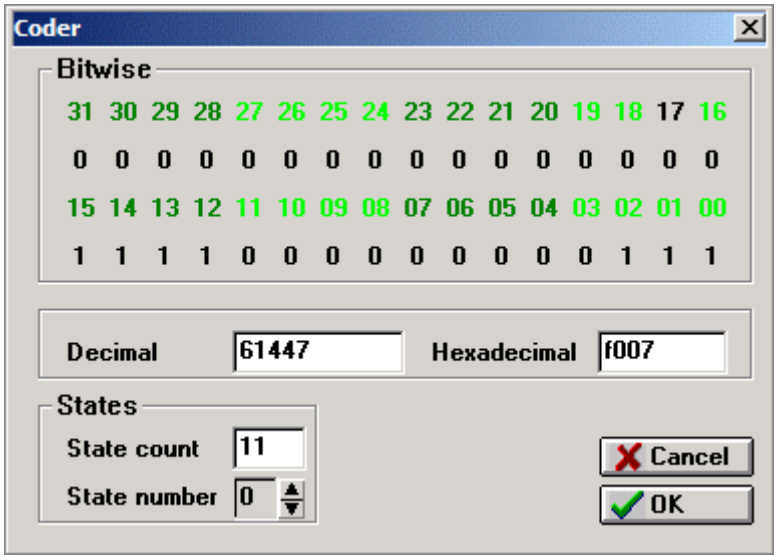


Figure 18-19. The 'Coder' Window.

The number of states recognized by the object is passed in *State count* field. This value may be also set in the main window for object defining. The state the code of which should be defined is selected in *State number* field. While setting the number in *Bitwise* frame, the current state coding is displayed - to change it, one should click with mouse on a bit value. Successive click operations cause change of the value in the following sequence:

- – bit value is unimportant
1 – bit has to be set
0 – bit has to be set to zero

Sate coding is automatically stored in the moment of choosing the other state.

The *Decimal* and *Hexadecimal* fields display decimal and hexadecimal value of defined state.

Selection frame - the check box that enables displaying an additional so called *selection frame* (border of the object) after the object has been selected while diagram refreshing. This frame allows to guess easy, which object has been selected as the last one.

By default, an object has only one state and is classified as static one. Transmission errors are alerted by change of color.

The object, while refreshing, can be selected by means of the mouse cursor or the *Tab* key (*Shift + Tab).* After having selected the object, the desired state of it (indicated by line color) can be selected by means of mouse cursor or the *Enter* key. Sending of the state is executed automatically after selection or acknowledge. Acknowledge can be carried out by striking the *Grey+* key or executing the appropriate action. You can cancel selection striking the *Esc* key or clicking the right mouse button.

Usage of the conversion function is possible: values of it are equal to indexes of individual colors, as well as the second conversion function that converts number of the object state to the value being sent (the above is referred to the option *with sending*). Decoding of the fixed-point monitored value (and coding of the sent values, if necessary) is also possible by means of one of the three strategies built-in into the object.

Natural Coding

The group of bits is checked. Number of bits is equal to the *number of states*, but number of the LSB defines "offset". The group of bits can consist only one value of "1". Number of the state is calculated as position of the "1" within the group of bits (the LSB position is number one). The zero state corresponds with the group of all zeros.

Binary Coding

The group of bits is checked. Number of bits is equal to the binary logarithm of *number of states*, but number of the LSB defines "offset". Number of state is calculated as content of the tested bit group.

User-Defined Coding

The whole word is checked. The "offset" parameter is ignored. The user assigns a string of ones and zeros (the *don't carry* states can be masked by "-") to each state. Number of state is calculated as the first of states, that the user assigned the same bits on the appropriate positions (regardless to the masked bits).

For the object, when parameter *with sending* has been set for it, the use of option *read first* is possible. In such a case the logical sum of the value being sent and the given value that has been read from the controller (the bits being controlled by the object are reset to zero) is written to the variable.

EXAMPLE

The object has four states and binary coding has been chosen. For encoding of four binary states (00, 01, 10, 11) the group of two bits is necessary in the word. The first (LSB) bit of the group determines content of the box named ***State - First bit***. If this contents is equal e.g. 4 that state only of the fifth and sixth bit influents on state of the object. The other bits are not referred to displaying the object what is being illustrated below.

----------xx----

("-" denotes 'don't carry' bits, but 'x' denotes the bit that influents on state of the object)

If you define the four lines that correspond with the four states mentioned above, e.g. yellow, green, blue and red ones (in this order), the yellow line correspond with the "00" state, the green one with the "01" state, etc. In case of writing, whether the red line are selected, the state corresponding with it, i.e. "11" on the 5^{th} and 6^{th} bits i.e. 48 in decimal, are sent to the variable. Whether the box ***read first*** is ticked the described procedure operates slightly different - the given variable is previously read, then its 5^{th} and 6^{th} bits are

reset to zero, next the bits that correspond with the color of the selected ellipse are put in on the place of those bits.

```
1000000000100000 / the variable that has been read
1000000000--0000  / the variable that has been read with "-" bits reset to 0
----------11----        / the bits set by the object
1000000000110000 / the variable that is to be sent
```

18.4.11. MESSAGES Object

Dimensions
Object Parameters
- Object Name
- File Name
- Text Delimiter
- Message List
- Selected Message
- Variable

An Example of *.msg File

The MESSAGES object enables displaying one of many specified texts on the diagram. As distinct from rather similar TEXTS object, it doesn't have the statuses number limitation. It is adapted for displaying big number of messages.

This object may be parametrized manually or from the text file, in which the texts and, possibly, their attributes are stored, if they are not default.

This is a dynamic object. The text, selected for displaying and its attributes (color, font, blinking) depend on monitored variable.

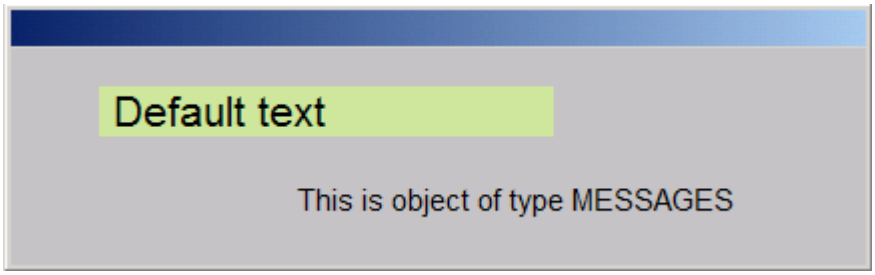


Figure 18-20. The MESSAGES Example.

Dimensions

Dimensions of MESSAGES object are defined with mouse. This area can't be scaled automatically, based on text's contents and the font used (like TEXTS object, for example). According to this, the designer is obliged to choose the correct area, for the all possible displayed objects could find enough room. The object itself can't do it, because the texts are not stored in the object, but in the text file and can be changed already after the parameterization. But, if some text is cut while displaying, then the object indicates this fact by placing the sequence of three dots "..." at the end of the text string.

MESSAGES object parametrization window:

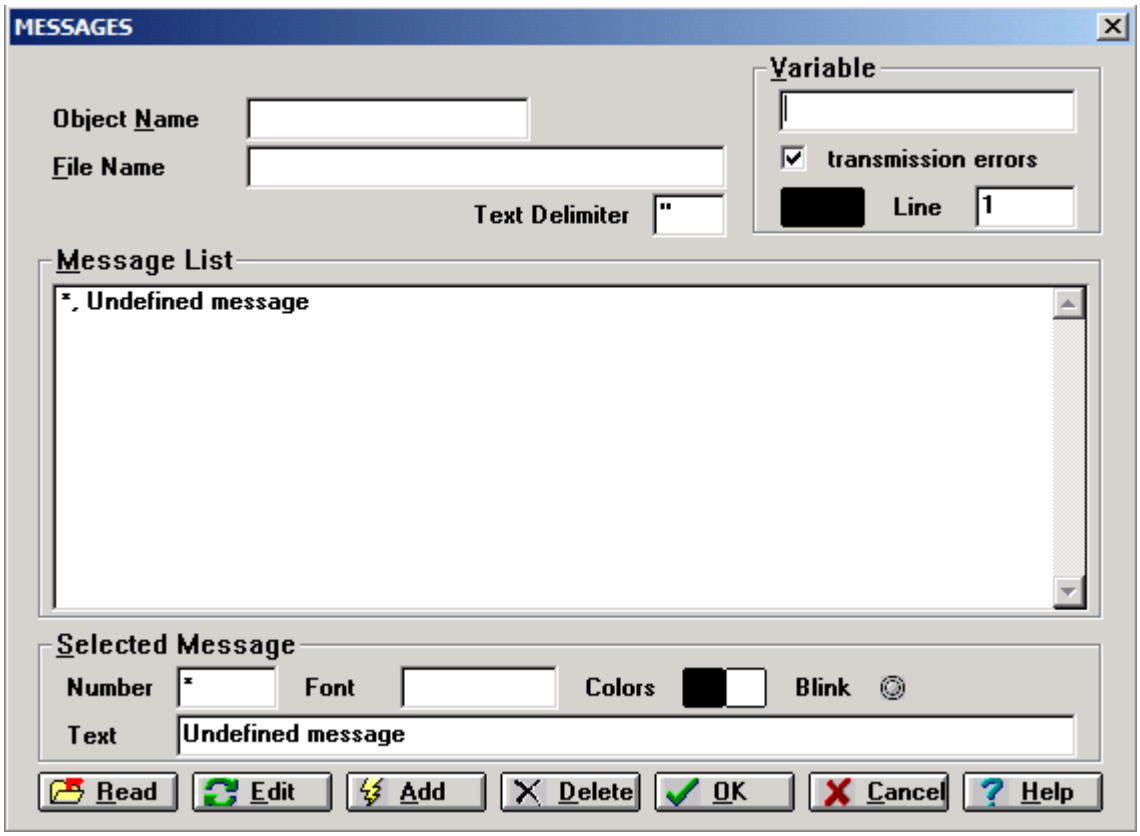


Figure 18-21. The MESSAGES Object Parameterization Window.

Parameters

Object Name - this text box is used for to put in the optional name of the object. If the name of the object will not be defined, its type together with coordinates will appear on the list of objects.

File Name - a text box for entering the filename with extension, containing the object definition. This file may be created manually or automatically. Syntax of this file has been described further. This file can be selected after pressing ***Read*** button. On reading, the default file extension is *.msg.

miter - a text box (1-character), it enables to declare a text delimiter (" – quotation marks by default) used in msg file.

Message List - a list of texts corresponding to particular object statuses with numbers of statuses. Elements of this list can be changed with ***Selected Message*** field group and buttons located underneath it.

Selected Message - a group of boxes displaying the text with attributes of the message selected in *Message*

List field. These fields can be edited and used for existing messages modification or inserting new ones.

Selected Message -Number **-** a text box enabling displaying and entering the number of displayed message. In this field, except for the number, also a range may appear, (e.g. **5..10**) a "*****" character(asterisk) symbolizing number of the default message .

Selected Message -Text **-** a text box enabling displaying and entering the text of current message (corresponding to the given status).

Selected Message -Font **-** a text box enabling entering the name of the font, that should be used for displaying the current text(corresponding to the given status). Not specifying the font causes accepting the default font (Dialog). Pressing the right mouse button at this field causes moving to the font selection window.

Selected Message -Colors **-** a color box enabling setting and displaying color of characters and background for displaying the text, corresponding to given status. After selecting this field, color selection window is displayed.

Selected Message -Blink. **-** a field signalling the mode of current text blinking (by changing the color). Clicking on this field causes entering the window, which enables defining the blinking mode.

Button ***Read*** **-** a button enabling selection and reading of the object definition from the text file with standard window, which allows opening the file. The filename is received from ***File Name*** field. If this field is empty, then *.msg is used.

Button ***Edit*** **-** a button enabling change of indicated message. All its parameters can be changed (contents, number, attributes). Particularly, the definition of default message can be changed, but then in the number field a "*****" character must be left. When modifying the number field, attention should be paid to not causing the numbers conflict.

Button ***Add*** **-** a button enabling adding new message, of characteristics defined with contents of ***Current message*** fields. When adding new message attention should be paid to not causing the numbers conflict. It is not possible to add the default message, except for the situation when the messages list is empty.

Button ***Delete*** **-** a button enabling deleting indicated message. The default message can't be deleted.

Variable - a text box enabling specifying the name of monitored data. Specifying this name is obligatory. By pressing the right mouse button in this field you can open the window with list of available variables and make a choice in it.

Transmission errors **-** a check box enabling signalling communication errors (object is crossed out). Default option.

Line - a color box of the line crossing out the object in case of occurrence of communication error and numeric field enabling specifying line's width. Selecting the color box causes displaying the color selection window.

After inserting new object on the diagram, *Message list* and *File Name* fields are empty. Proper messages should be entered then. There are two possibilities of filling out this list. All the messages can either be entered into the text file with set syntax, or entered directly from the parametrization window and saved.

The first solution is better in case of first parametrization, and the second one is convenient for making small corrections.

Below, the syntax of the file defining the MESSAGES type object is shown. These files have ***.msg** extension.

The syntax of single line is as follows:

Number, Text[,Font[,Color[,Background[,Blinking[,Color1[,Color2]]]]]]

where:

Number	- is the number specifying the text to be displayed, if the variable assumes its value; decimal and hexadecimal notation is accepted, e.g. in the form **0x10**; it is possible to specify the value range e.g. **10..20**. This means if the monitored variable assumes the value from within the specified range, the text defined for it will be displayed; the ***** character in this field means the default text; numbers must be from within the range **1..n**;
Text	**-** is any text, delimited with **""** marks (by default the quotation mark , with possibility of changing);
Font	**-** is optional name of given text's font;
Color	**-** is optional characters color;
Background	**-** is optional background color;
Blinking	**-** is optional name of blinking mode; following shortenings has been accepted for particular modes: "NO", "X", "FX", "BX", "F", "B", "FB";
Color1	**-** is a color required for blinking with characters;
Color2	**-** is a color required for blinking with background;

First line should contain default message. The **#** mark at the beginning of the line means a comment. Empty lines are accepted.

For encoding the colors, a notation used in **asix** for writing the colors into the trends definition file has been used. The color is saved as a string:

RGB-*rrr***-***ggg***-***bbb*

Where:

rrr	- red color saturation (0..255);

ggg - green color saturation (0..255);
bbb - blue color saturation (0..255).

Color 1 and Color2 fields must be present, if the suitable blinking mode is active ("F", "B" or "FB"). Particular optional fields may be empty. In such case, the default field value is assumed.
In most cases the optional fields will not be used and usually they will not be filled out with the editor. Principally, these fields will be generated automatically while exiting the object editing, after the user specifies, that e.g. the text No. 100 should be red and additionally blinking.

Every file with messages definition must be checked before reading. Following errors are detected:
- wrong field contents, e.g. should be a text, and is a color,
- multiple definition of given message, e.g. two lines of type
  1,"text....
  1,"text....
- overlapping areas, e.g. lines of type
  1..30, "text....
  20..50, "text....
- empty areas e.g.
  20..10, "text.....
- lack of the line with default message

Below is an example of *.msg file.

EXAMPLE

```
*,"Default text ...",Dialog,RGB-0-0-0,RGB-0-255-128,
1,"Text displayed for status 1",Dialog,RGB-0-0-0,RGB-255-255-128,
3..10,"Text displayed for statuses from 3 to 10",Dialog,RGB-0-0-0,RGB-
255-0-0,FB,RGB-0-0-0,RGB-255-128-64,
```

After inserting a new MESSAGES object on the diagram, if we quit on manual filling the definitions file out, a ***Read*** button should be pressed, to read pre-prepared text file with the definition. This file may have only the positions defining the number and the text filled out. In this file has no errors, then the ***Messages list*** will be filled out.

Having messages list prepared, you can edit, add and delete messages. For this purpose ***Change***, ***Add*** and ***Delete*** buttons are used**,** and also the ***Current message*** field. To edit specified message it has to be pointed on the list first. Then its number (numbers), contents, and attributes will appear in suitable fields of ***Current message*** group. Then they can be edited, and after the editing the ***Change*** button should be pressed. Adding and deleting messages looks similarly.

You should remember, that the default message is handled in a special way. It cannot be deleted, a second one can't be added – you can only edit it without changing the number field (it is read-only then).

18.4.12. MOVE CONTROLLER Object

MOVEMENT CONTROLLER object allows for animation the movement of technological mask objects. The movement is executed in axes OX an OY and can pertain to one or many objects at the same time.

Parameters

Object Name **-** a text box enabling entering an optional object name;

Vertically – Variable Name - a text box declaring the variable on which the movement of controlled objects in axes OY depends on;

Vertically – Position - sets in pixels a vertical range for movement of controlled object;

Vertically – Range - sets a range for variable values;

Horizontally – Variable Name - a text box declaring the variable on which the movement of controlled objects in axes OX depends on;

Horizontally – Position - sets in pixels a horizontal range for movement of controlled object;

Horizontally – Range - sets a range for variable values;

All Objects - displayes the list of all named objects put on visualization mask.

Controlled Objects - displayes the list of all objects controlled by MOVEMENT CONTROLLER.

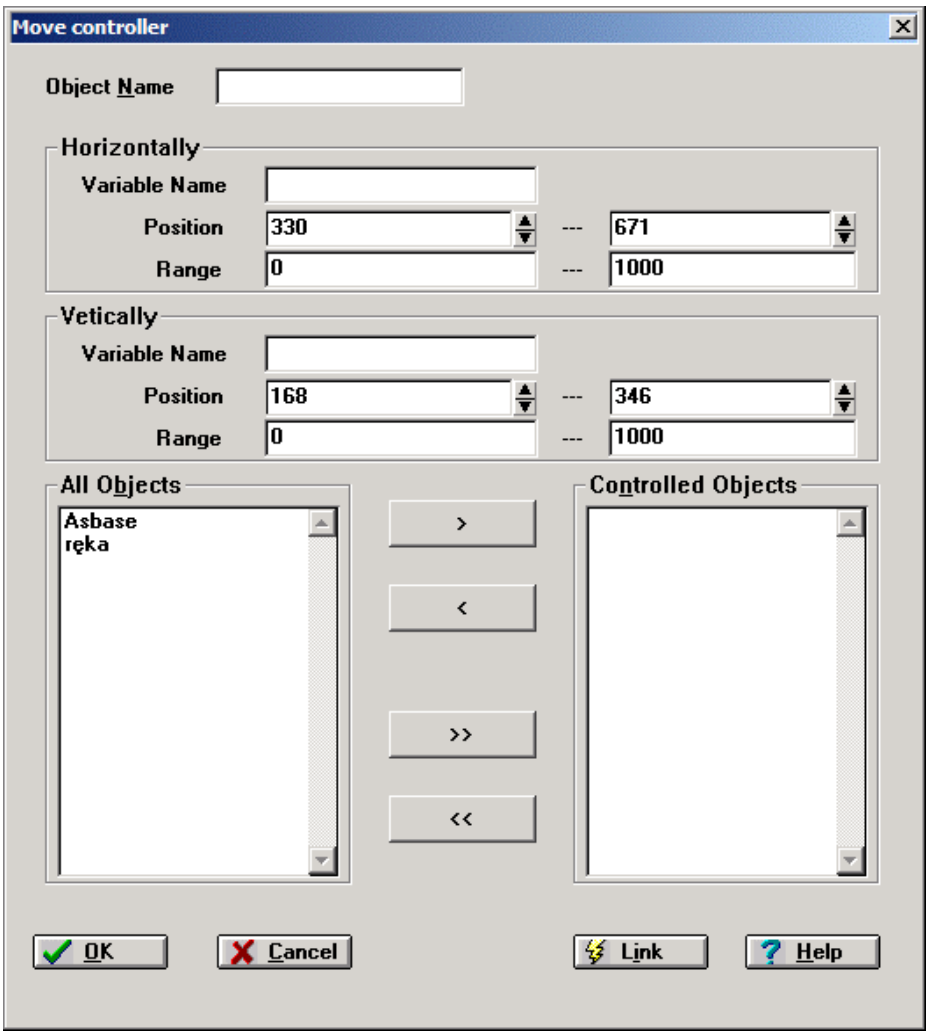


Figure 18-22.The MOVEMENT CONTROLLER configuration window.

18.4.13. MOTOR Object

Dimensions
Object Parameters
Object Name
Blocade
Control
Font
Variable
Control Field
FORMAT OF MONITORED VARIABLE
FORMAT OF CONTROLLED VARIABLE

The MOTOR object enables displaying of the motor status on the diagram and driving of it as well. Usually such objects are grouped into so called motor tables. The object is adjusted for collaboration with the input and output words of the appropriate format (forced by software of the controller). It is the both dynamic and selectable object. Status of the object is displayed by means of three windows coupled with the buttons and the fourth one used for electric failure indication and technological turning off. The other three windows are

used for selection of location, mode and type of driving. Sending of controls can be protected by password.

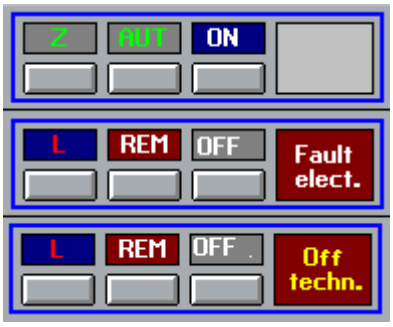


Figure 18-23. Examples of MOTOR type objects.

Dimensions

Coordinates of the MOTOR object are fixed and cannot be changed.

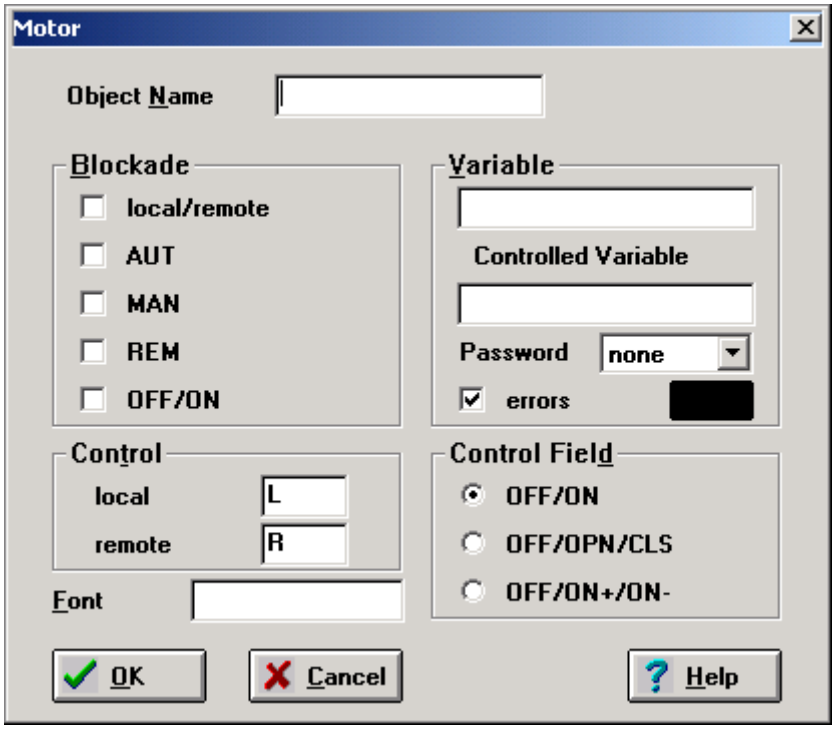


Figure 18-24. The MOTOR Object Parameterization Window.

Parameters

Object Name - this text box is used for putting inti it the optional name of the object. If the name of the object will not be defined, its type together with coordinates will appear on the list of objects.

Blockade - local/remote - a check box that enables locking the feature of switching by means of the shown control panel the local mode into remote one and vice versa.

Blockade - AUT - a check box that enables locking the feature of switching on, by means of the shown control panel, the automatic control mode.

Blockade - MAN - a check box that enables locking the feature of switching on, by means of the shown control panel, the manual control mode.

Blockade - OVR - a check box that enables locking the feature of switching on, by means of the shown control panel, the overhaul mode.

Blockade - OFF/ON - a check box that enables locking the feature of switching, by means of the shown control panel, the motor on and off.

Control - Local - a text box that enables declaration of abbreviation that stands for local control mode. By default it is "L" letter.

Control - Remote - a text box that enables declaration of abbreviation that stands for remote control mode. By default it is "R" letter.

Font - a text box that enables to enter the name of the font that is going to be used for displaying text information within the *MOTOR* object. If the font name is omitted, the default font (Dialog) is assigned. Clicking the right mouse button causes displaying the font selection window. Because of fixed dimensions of the object one should constrain to the low height fonts (*Small, Dialog etc.*).

Control Field- OFF/ON - a radio button that enables displaying the strings "ON" and "OFF" in the appropriate box of the object (e.g. for motors).

Control Field- OFF/OPN/CLS **-** a radio button that enables displaying the strings "OFF", "OPN" and "CLS" in the appropriate box of the object (e.g. for valves).

Control Field- OFF/ON+/ON **-** a radio button that enables displaying the strings "OFF", "ON+" and "ON-" in the appropriate box of the object (e.g. for two-way motors).

Variable - a text box that enables declaration of the name of the monitored variable. Declaration of this name is obligatory. Clicking the right mouse button on this box you can open the window with the list of available variables and make selection within this window.

Control variable - a text field, which allows specifying the name of control variable. Specifying this name is necessary. By pressing the right mouse button in this field you can open the window with the list of available variables and make the choice in it. It is possible to specify the name of controlled data, using notation with **#** character. If in the field of variables name the #_ characters with suffix are entered, then during the application's run it will be treated as

the variable of name containing the name of monitored variable instead of the # character.

Password - a password combo box that protects sending of the controls (four levels of password can be selected, no password is the default setting).

Errors **-** a check box that enables monitoring the communication errors and the corresponding color box (the object changes color of the border). The option is set by default.

All the colors appearing in the object are predefined. Operation states of the motor are displayed in dark-gray (the overhaul mode state is an exclusion - it is displayed in red box). The selected states are displayed in boxes with blue background. The locked states are displayed in boxes with violet background. The object has black border that can change its color in case of errors. The border changes also its color due to indicate selection. The color different from the border color and selection color should be chosen for error indication.

The object, while refreshing, can be selected by means of the mouse cursor or the *Tab* key (*Shift + Tab).* After having selected the object, the desired state of the motor can be selected by means of clicking with the mouse cursor appropriate buttons. Striking the *Grey+* key or executing the appropriate action carries out sending of the state. You can cancel selection striking the *Esc* key or clicking the right mouse button. Striking (multiple times) the Enter key can carry out selection of the state.

Format of Monitored Variable

bit 0 - state OFF/CLOSED
bit 1 - state ON/ON+/OPEN
bit 2 - state ON -
bit 3 - technological breakdown
bit 4 - electric failure
bit 5 - overhaul mode OVR
bit 6 - manual mode MAN
bit 7 - automatic mode AUT
bit 13 - 0 - remote control, 1 - local control

Format of Controlled Variable

bit 0 - OFF/CLOSE pulse
bit 1 - ON/ON+/OPEN pulse
bit 2 - ON- pulse -
bit 3 - selection of remote control
bit 4 - selection of local control
bit 5 - turning on the overhaul mode OVR
bit 6 - turning on the manual mode MAN
bit 7 - turning on the automatic mode AUT

The LSB (less significant bit) is the bit number 0.

18.4.14. NUMBER Object

Dimensions
Object Parameters
- Object Name
- Colors and Fonts
- Format
- Range
- Limits
- Parameters
- Variable
- Justify
- Vertically

Operation Description
Format Description
Configuring from the VarDef

The NUMBER object enables displaying the number that is corresponding with the system variable on the diagram. It is the dynamic object. As option, it can be the selectable object. In this case, it enables writing to the object the value that will be later written to the system variable. Value sending can be protected by password. The NUMBER object can change its color and/or blinking depending on cases, when exceeding critical values or some logic conditions occur.

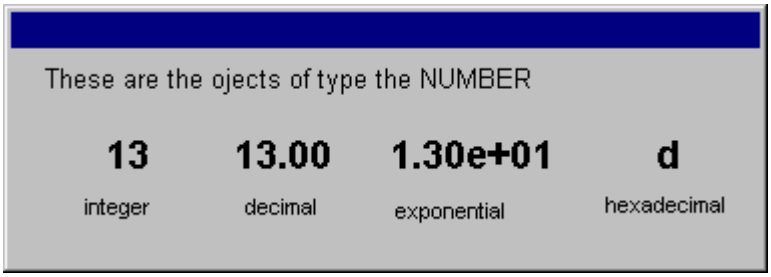


Figure 18-25. The NUMEBR Example.

Dimensions

Height and width are set by automatically accordingly to the specified format for the maximum long string being result of the given range. Dimensions of the object can be modified by means of the mouse. If the check box ***Area by Format*** is ticked dimensions will be automatically modified to match the adopted format.

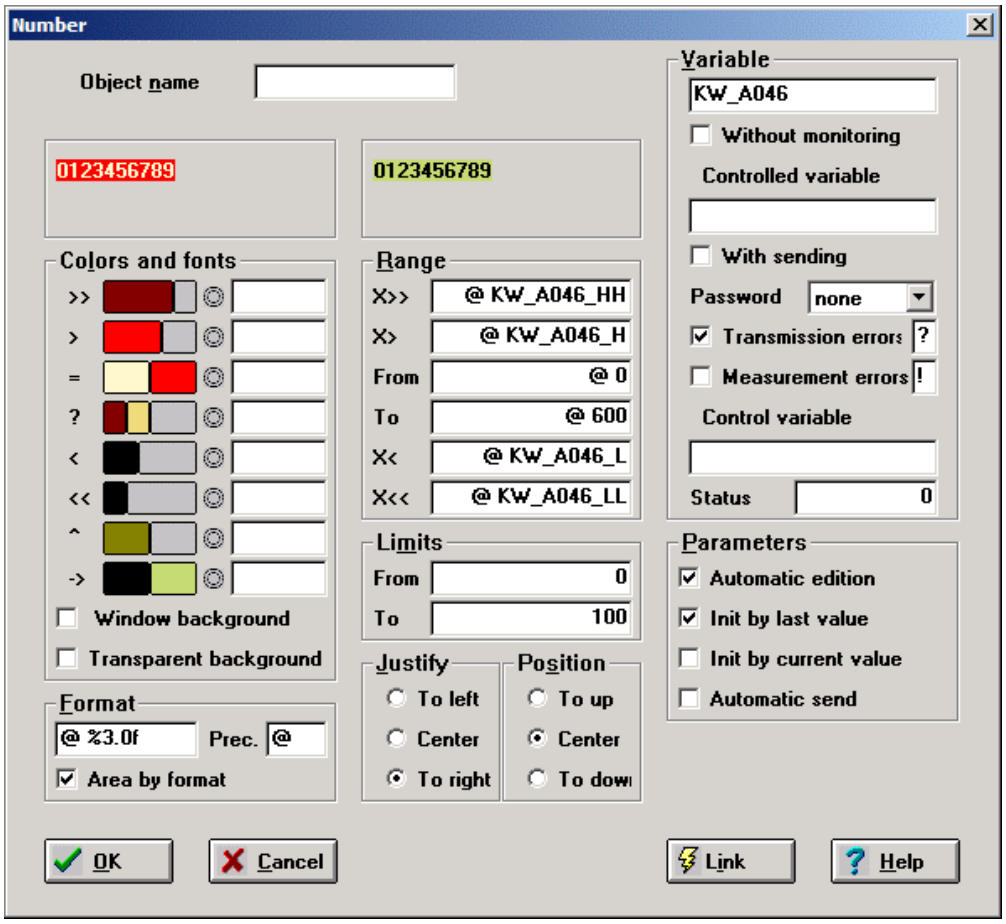


Figure 18-26. The NUMBER Object Parameterization Window.

Parameters

Object Name - this text box is used to put into it the optional name of the object. If the name of the object will not be defined, its type together with coordinates will appear on the list of objects.

Colors and Fonts->> - a color box enabling setting of the color of digits (the left box) and background (the right one) when the upper alarm limit is exceeded.

Colors and Fonts -> - a color box enabling setting the color of digits (the left box) and background (the right one) when the upper warning limit is exceeded.

Colors and Fonts -= - a color box enabling setting the basic color of digits (the left box) and background (the right one) of the displayed number.

Colors and Fonts -? - a color box enabling setting the color of digits (the left and central boxes) and background (the right one) of the displayed number in case of communication error (the left box) and measurement error (the central one).

Colors and Fonts -< - a color box enabling setting the color of digits (the left box) and background (the right one) when the lower warning limit is exceeded.

Colors and Fonts -<< - a color box enabling setting the color of digits (the left box) and background (the right one) when the lower alarm limit is exceeded.

Colors and Fonts -^ - a color box enabling setting the color of digits (the left box) and background (the right one) while it is being edited that precedes sending of the value.

Colors and Fonts - -> - a color box enabling setting the color of digits (the left box) and background (the right one) after editing is completed, but before sending of the value in mode of sending with acknowledge.

Colors and Fonts - Window background - a check box that enables to force drawing the window background the same as window color.

Colors and Fonts - Transparent background - a check box that enables to force drawing a transparent background.

Colors and Fonts - to the right from particular limits' color selection fields, blinking attribute indicators are located. Further to the right is a column of fonts selection field. The fonts may be selected individually for each of the limits, which enables e.g. declaring bigger font for alarm situations, to make it easier for the Operator to notice exceeding the technological limits.

Format - a text box enabling declaration of the format of output number. The allowed formats are originated from the C language and will be further discussed. By default the *%d* format (integer number) is adopted.

Format – Prec. **-** a numeric box that enables setting the precision. For example, entering the number 10 in this box causes that the number will be rounded to 10 range. By default value of this box is set to 0 what denotes no rounding.

Format - Area by Format **-** a check box that enables forcing that area occupied by the object will match the declared format and font, e.g. for the *%5u* format it denotes, that the occupied area will match the five digits with dimensions defined by the selected font.

Range -X>> - a text and numeric box that enables entering either the numeric value of upper alarm limit or name of the variable that contain this limitation. By default that value is equal 90. In case, when the name of variable is declared, the logical condition can be additionally defined in the box located on the right. The possible conditions will be described in further part.

Range -X> - a text and numeric box that enables entering either the numeric value of upper warning limit or name of the variable that contain this limit. By default that value is equal 75. In case, when the name of variable is declared, the logical condition can be additionally defined in the box located on the right. The possible conditions will be described in further part.

Range - from - a numeric box that enables entering the numeric value of lower range. By default that value is equal 0. The lower numbers will be displayed. The value is used exclusively for definition of the area occupied by the object.

Range - to - a numeric box that enables entering the numeric value of upper range. By default that value is equal 100. The higher numbers will be displayed. The value is used exclusively for definition of the area occupied by the object.

Range -X< - a text and numeric box that enables entering either the numeric value of lower warning limit or name of the variable that contain this limit. By default that value are equal 25. In case, when the name of variable is declared, the logical condition can be additionally defined in the box located on the right. The possible conditions will be described in further part.

Range -X<< - a text and numeric box that enables entering either the numeric value of lower alarm limit or name of the variable that contain this limit. By default that value is equal 10. In case, when the name of variable is declared, the logical condition can be additionally defined in the box located on the right. The possible conditions will be described in further part.

Limits - from - a numeric box that enables definition of the numeric value of lower limit of the entered numbers (it concerns the option ***with sending***). By default that value is equal 0. The less numbers will not be received and sent.

Limits - to - a numeric box that enables definition of the numeric value of upper limit of the entered numbers (it concerns the option ***with sending***). By default that value is equal 100. The greater numbers will not be received and sent.

Parameters - Automatic Editing **-** a check box, that enables editing of the number just after it has been selected (clicked). Otherwise, the object should be selected and then you have to press the *Enter* key to start editing.

Parameters - Init by Last Value **-** a check box that enables initiation of the number box before editing with the previously entered value.

Parameters - Init by Current Value **-** a check box that enables initiation of the number box before editing with the current value.

Parameters - Automatic Send - a check box that enables immediate sending the numeric value after having the editing completed by striking the *Enter* key. Otherwise, the number is sent after having pressed the "*Grey+*" key or having executed the appropriate action.

Variable - a text box that enables entering the name of measured data. Entering of that data is not obligatory, whether the option ***without***

monitoring has been selected. By clicking the right mouse button within this box you can open the window with the list of available variables and carry out selection among them.

Controlled Variable - a text box that enables entering the name of the control variable. Entering of that data is obligatory, whether the option ***with sending*** has been selected. By clicking the right mouse button within this box you can open the window with the list of available variables and carry out selection among them.

Without monitoring **-** a check box that that enables setting parameters of the object in such a manner, that the displayed status is not dependent on any system variable and will serve only for indication of the value that is to be sent.

With sending **-** a check box that enables setting parameters in such a manner, that the value corresponding with the state of the object would be sent to the system variable. Sending of value is executed after having the object selected e.g. by mouse clicking, the further process depends on the mode of sending.

Password - a combobox of the password that protects sending of the controls (four levels of password can be selected, no password is the default setting).

Transmission error **-** a check box that enables warning at transmission errors occur. In such a case the sign placed in the box on the right will be added to the number, by default "**?**". Placing of two "**?**" characters in this box causes, that the whole number box will be filled with question marks whether this error occurs. The option is set by default.

Measurement error **-** a check box that enables warning at measurement errors occur. In such a case the sign placed in the box on the right will be added to the number, by default "**!**". Placing of two "**!**" signs in this box causes, that the whole number box will be filled with exclamation marks whether this error occurs. For this purpose the ***Control Variable*** must be specified.

Control Variable **-** a text field which enables specifying the name of control data. Specifying this name is necessary, if ***measurement errors*** option has been chosen. By clicking the right mouse button in this field you can open the window containing the list of available variables and make a choice in this window. It is also possible to specify the name of the control data by using notation with **#** character. If we enter the #_ characters with suffix in the variable name field, then during the application's run it will be treated as a variable,

	with name containing the name of monitored variable instead of the # character.
Status	- a text box that enables entering of the hexadecimal mask. If logic product (conjunction) of that mask and the check data is different from zero, the measurement error is displayed.
Justify - To Left	- a radio button that forces displaying the number in the object box justified to left.
Justify - Center	- a radio button that forces centering the number in the object box.
Justify - To Right	- a radio button that forces displaying the number in the object box justified to right.
Position - to up	- a radio button that forces vertical justification of the number to the upper edge of the object's field.
Position - center	- a radio button that forces vertical centering the number inside the object's field.
Position – to down	- a radio button that forces vertical justification of the number to the lower edge of the object's field.

NOTICE Empty texts and the texts consisted of white spaces are acceptable in limit fields and using them is treated as abandoning verification.

Operation Description

The following formats are allowed: %i, %d, %u, %o, %x, %li, %ld, %lu, %lo, %lx, %f, %e, %g (integer number with and without sign, octal and hexadecimal numbers and also floating point numbers). The box length may be defined for all the possible formats. The fraction length (number of fractional digits) can be defined for floating point formats %f, %e and %g. When the error occurs the check mark (by default ? or !) is to be added to the number or the whole number will be replaced with the string of check marks. Room for the check mark should be provided while the format is being specified.

The specified format is obligatory both for input operations (monitoring) and output ones (control). In case of control, if automatic editing has been selected, the editing mode is initialised after having pointed an object by means of the cursor. Otherwise, pointing an object with the cursor causes only change of the cursor into the editing one, but editing itself is started only after the *Enter* key has been struck. The last sent or last received number can appear in the editing box (as selected). The number of entered characters is limited to width of the window if the above has been defined while specifying the format otherwise it is limited to 20 characters. Terminating the editing by means of the *Enter* key can cause immediate sending of the number, however, the number can also be sent after having pressed the *Grey+* key or after execution of the defined action. In such a case the number is displayed in color indicating that the input of the value has been completed and it is waiting for to be sent. While editing is in progress and waiting for sending value of the measured variable is not displayed. Pressing the *Esc* key or clicking the right mouse button will cause canceling the editing.

The displayed number can change its color in cases of exceeding variable limits. The exceeding limits can be specified as numbers (constants) or names

of system variables. The attempt of entering and sending the incorrect number or value out of limits of controls evokes the error prompt. When type of the system variable does not match the type specified for format declaration, automatic conversion is carried out (that conversion may be reason of errors - lost of fractional part or overflow). This is why type of the number should rather match type of the variable.

If overflows are specified as names of variables the logical conditions related to variables can be defined in adjacent boxes (right of names). If the condition is true the color of displayed number is changed. The logic condition is a string that consists of the hexadecimal number (condition) together with preceding sign "&" or "|". For the first case (&) the condition is true if the logical product (conjunction) of the variable and condition is equal to the condition, for the second case (|) the condition is true if the logical product (conjunction) of the variable and condition is different from zero. In other words, for the first case all the bits of the condition are required to be set for the given variable, for the second case it is enough if only one bit of the condition (at least) is equal to one. The sign "|" can be omitted while the condition is being specified.

EXAMPLE

NUMBER object should change its color in case, when the 2^0 and 2^2 bits will be set for value of the "X" variable. In such a case in the box ***Range - >>*** one should put in consecutively "X" and "&5" in the condition box.

Description of Formats

The format parameter has to be compatible with the following syntax:

[%][[0]width][.precision][date_format][time_format][l|h]type

where:

%	- optional beginning of a format;
type	- formatting method:
d,i	- decimal signed integer,
u	- decimal unsigned integer,
o	- octal number,
x	- hexadecimal number using small letters abcdef,
X	- hexadecimal number using capital letters ABCDEF,
B	- number in BCD code,
F,e,g,E,G	- floating-point number,
D	- date, input datum is treated as a number of seconds since 1970-01-01-00:00, 32 bites;
T	- time, input datum is treated as number of seconds, 32 bites;
[0]width	- minimum field width; adding the 0 before the number of width means that the field will be completed by the 0 figures on the left; the element is used only for d,i,u,o,x,X,B,f,e,g,E,G formats;
precision	- quantity of fractal digits for f,e,g,E,G;
l/h	- additional input data type definition: 1-32 bits, h – 16 bits; only for d,i,u,o,x,X,B formats;
data format	- detailed date format definition used only for D; it consists of sequence of characters defining separators and date components in arbitrary sequence; there are the following components: y-year, m-month, d-day, h –hour,

time_format	n-minute, s-second; no date format definition causes formating acording to the pattern: y/m/d h:n:s; - detailed tme format definition used only for T; it consists of sequence of characters defining separators and time components in arbitrary sequence; there are the following components: d-days, h –hours, n-minutes, s-seconds, z-milliseconds; no time format definition causes formating acording to the pattern: h:n:s;

NOTICE If the field *d* is used, then the field *h* are not higher then 23, e.g. if the format "h:n:sT" displays „46:10:00", the format „d h:n:sT" displays „1 22:10:00".

NOTICE You may set the letter "I" before the letter "T"– that means that an input datum is a number of seconds (and not milliseconds).

NOTICE For D and T formats it is possible to set an empty text that corresponds to 0, e.g.:
1970-1-1 0:0:0 for D,
0:0:0 for T.

The concept of formats is derived from the C programming language.

EXAMPLES

%i - a fixed-point number, all significant digits are specified.
%7lu - a long fixed-point number, without sign, seven positions.
%6.2f - a floating-point number, decimal, six positions, two decimal positions.
h:n:s:zT - time format with milliseconds.
T - there is no neeed to put % before T.
y/m/dD - date format without time.
y-m-dD - date format of the following form: 2005-11-30.

Configuring from the Variable Definitions Database

The NUMBER object is provided with features that allow setting the following parameters from the Variable Database:

- format according to the type of number displayed e.g. **%f**, **%i** etc.
- range of variation, e.g. 0, 100, according to the name of displayed variable,
- limits, exceeding of which will trigger the change of color of **Limit**, **Limit&10** variables names **(**name of variable with the bit mask**)** according to the name of displayed variable,
- ranges of sent values, for example 0, 100, according to the name of control variable,
- name and mask of the control data.

In this purpose NUMBER object has been additionally provided with **link** button, pressing of which causes inserting the contents of given field to the base, preceded with **@** character, into all positions, that may potentially be set from the variables base.

In the example below pressing the Link button has caused printing the contents of the fields from the base for:

- critical maximum limit – value 100;
- maximum limit – variable named A02_H;
- minimum limit – variable named A02_L;
- critical minimum limit – value of the KW_A110_S variable bit, defined by the mask(in this case hexadecimal 4);
- control data – variable A02?;
- control data's mask – 4;
- number format.

See: ***Figure 18-26. The NUMBER Object Parameterization Window.***

In order to fill the field from the base, it must be empty in the moment of clicking the *Link* button.

For the controlled and control variable, appropriate selection field must be selected.

18.4.15. OBJECTS CONTROLLER Object

OBJECTS CONTROLLER allows for visibility control of the technological mask objects. 'Visibility' mechanism pertains to objects of all classes. A hidden (invisible) object is not displayed and does not perform any functions. In particular, it is possible to hide BUTTON-type object or to disable CALCULATOR object.

Controlling of the object visibility status is not declared in the object itself, but is executed with use of OBJECTS CONTROLLER object class, which can control the visibility of other objects on the basis of process variable state or the currently set password level (which also co-operates with the user log-in systems - the internal one and the one existing in AsAudit module).

Parameters

Control by Password Level - controlled objects are visible only for the operators who have a password level equal or higher than the established level.

Control by Variable's Value - consists in declaring bit mask of the variable on which displaying controlled object depends. When the condition is performed, the object is invisible or invisible (it depends on parameters: *Hide* / *Show*).

Using both types of control causes the following object operation: if an operator has the proper password level, objects are visible; additionally, controlled objects become visible/invisible if the condition of control by variable's value is performed.

Control by Mouse Position - object becomes visible if an operator moves the mouse cursor over OBJECTS CONTROLLER.

Controlled objects are connected with OBJECTS CONTROLLER with use of object names entered in *Controlled Objects* field of OBJECTS CONTROLLER

configuration window. Objects can have the same name that allows connecting many objects to OBJECTS CONTROLLER using only one name.

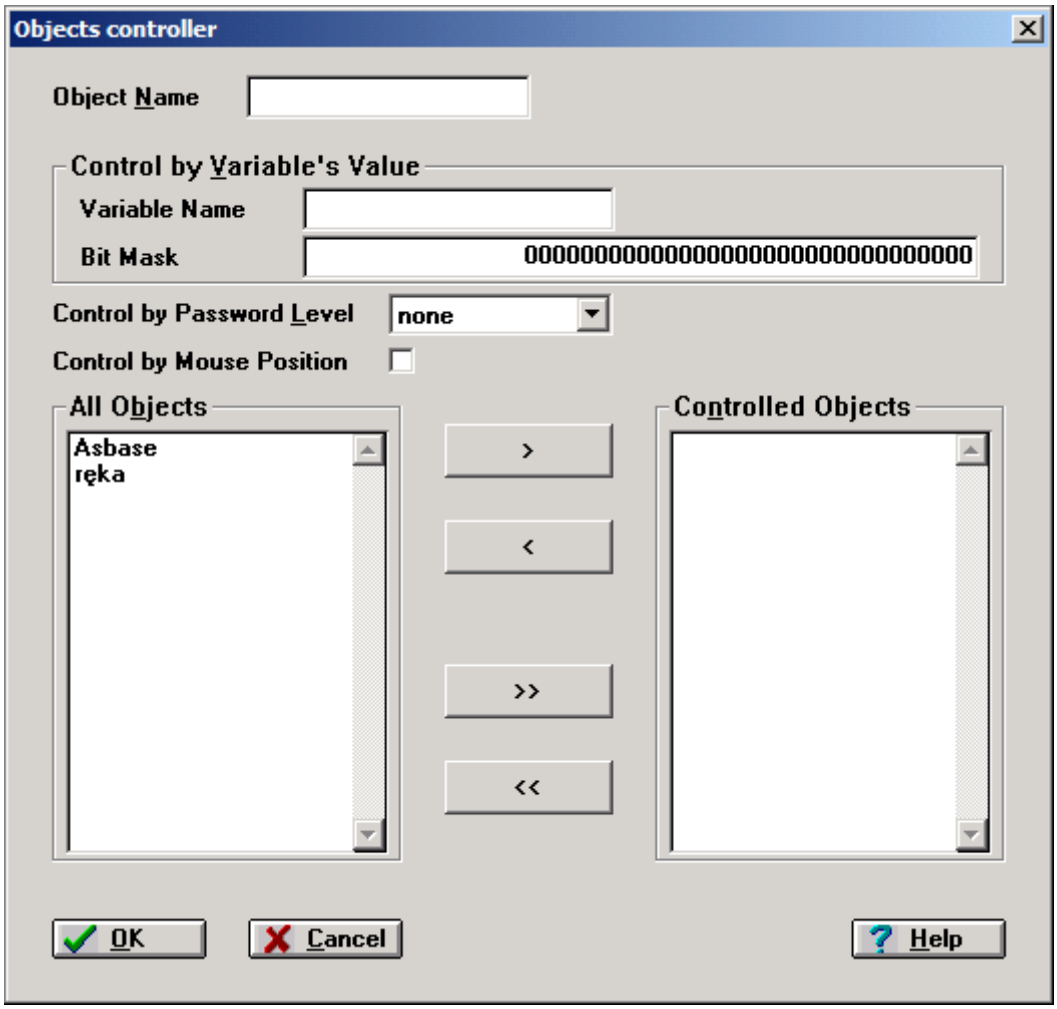


Figure 18-27. The OBJECTS CONTROLLER Configuration Window.

18.4.16. STRING Object

Dimensions
Object Parameters
Object Name
Parameters
Editing
Justification
Variable
Configuring from the VarDef

The STRING object enables displaying on the diagram the string (table) of character, i.e. the text. The text should be assigned to the variable of ASMEN (it can come from the controller). Whether the parameter ***With Sending*** has been set for the STRING object, the object itself can be used for sending of a string of character to a variable of the ASMEN (and, in further, to the

controller). This object collaborates with the special conversion function NOTHING_TEXT. The object usually collaborates with variables that have not assigned physical transmission channels, but only the virtual one (NONE protocol). Then, it is used for displaying and setting of text-type fields of database (cf. *XBASE*). Sending of controls can be protected by password.

On the diagram, there are no differences between this object and the objects of TEXT or TEXTS type.

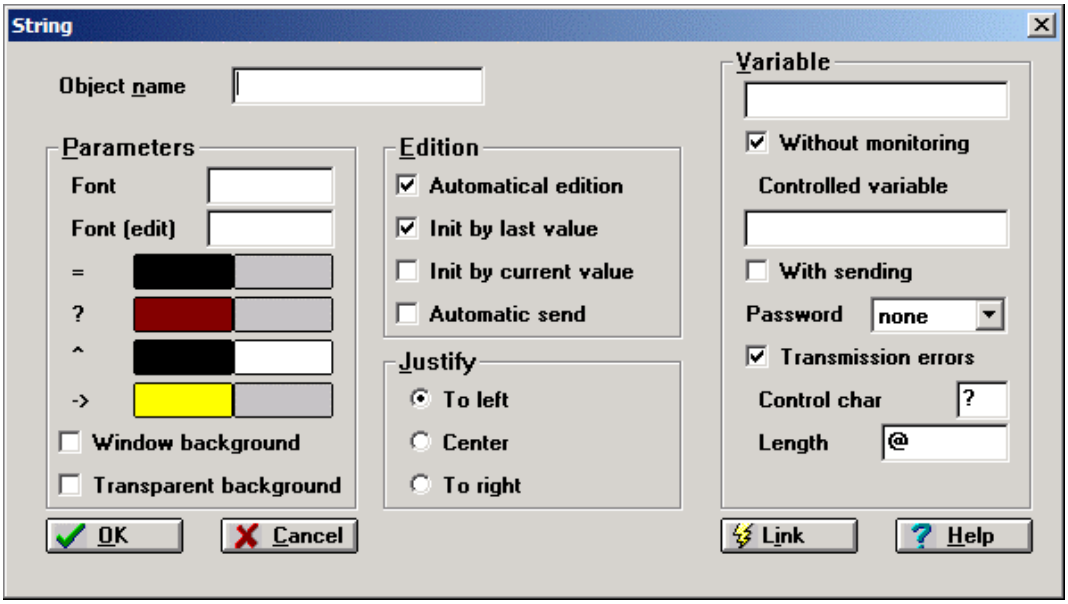


Figure 18-28. The STRING Example.

Dimensions

Dimensions of the STRING object are determined by the maximum length of the string and the applied font. The area defined, as above is the minimum and can be enlarged by means of the mouse. The attempt to reduce dimensions of the object below the minimum will be unsuccessful.

Maximum length of the text declared for the STRING object should match the appropriate parameter of the NOTHING_TEXT conversion function.

In case of control selection of automatic editing causes, that after having the object pointed with the cursor the editing mode is automatically initialized. Otherwise, pointing an object with the cursor causes only change of the cursor into the editing one, but editing itself is started only after the *Enter* key has been struck. The last sent or last received text can appear in the editing box (as selected). Terminating of editing by means of the *Enter* key can cause immediate sending of the text, however, the text can also be sent after having pressed the *Grey+* key or after execution of the defined action. In such a case the text is displayed in color indicating that the input of the value has been completed and it is waiting for to be sent. While editing is in progress and waiting for sending value of the text value of the monitored variable is not displayed. The attempt to put in and send the incorrect (too long) string brings the effect of prohibition of input (if you tries to put in the text longer than the parameter "length" allows) or the text will be cut off (if length of the text that

is to be sent is greater than the appropriate parameter of conversion function allows).

Parameters

Object Name - this text box is used for to put in the optional name of the object. If the name of the object will not be defined, its type together with coordinates will appear on the list of objects.

Editing - Automatic Editing **-** a check box that enables editing of the string just after it has been selected (clicked). Otherwise, for to start editing the object should be selected and then you have to strike the *Enter* key.

Editing - Init by Last Value **-** a check box that enables initiation of the text string box before editing with the previously entered text.

Editing - Init by Current Value **-** a check box that enables initiation of the text string box before editing with the currently displayed text.

Editing - Automatic Send - a check box that enables immediate sending of the text (table of characters) after having the editing completed by striking the *Enter* key. Otherwise, the text is sent after having pressed the "*Grey+*" key or having executed the appropriate action.

Variable - the text box that enables entering the name of monitored data. Entering of that data is not obligatory, when the option ***without monitoring*** has been selected. By clicking the right mouse button within this box you can open the window with the list of available variables and carry out selection among them.

Controlled Variable - a text box that enables entering the name of the controlled variable. Entering of that data is obligatory, whether the option ***with sending*** has been selected. By clicking the right mouse button within this box you can open the window with the list of available variables and carry out selection among them.

Without monitoring **-** a check box that that enables setting parameters of the object in such a manner, that the displayed status is not dependent on any system variable and will serve only for indication of the value that is to be sent.

With sending **-** a check box that enables setting parameters in such a manner, that the value corresponding with the state of the object would be sent to the system variable. Sending of value is executed after having the object selected e.g. by mouse click, the further process depends on the mode of sending.

Password - a combo box of the password that protects sending of the controls (four levels of password can be selected, no password is the default setting).

Transmission error **-** a check box that enables warning at transmission errors occur. In such a case the character placed in the box ***Control Char*** will be added to the text (by default "**?**"). The option is set by default.

Control Char - a text box that enables declaration of the character. When the transmission error occurs, this character will be added to the text.

Length - a numeric box that defines maximum length of a text that constitutes the string. By default, this box is set to 20. Please to remember about correlation of that number with the appropriate parameter of the NOTHING_TEXT function.

Parameters - Font - a text box that enables entering the name of the font that is to be used for displaying the string. If no font is indicated, the default font (*Dialog*) will be adopted. By clicking the right mouse button within this box you can open the font selection window.

Parameters –Font (edit) - a text box that enables entering the name of the font that is to be used for displaying the string while editing and sending the value. If no font is indicated, the default font (*Dialog*) will be adopted. By clicking the right mouse button within this box you can open the font selection window.

Parameters = - a color box enabling setting of the color of string (the left box) and background (the right one) of the displayed text.

Parameters –? - a color box enabling setting of the color of string (the left box) and background (the right one) when the communication error occurs.

Parameters –^ - a color box enabling setting of the color of string (the left box) and background (the right one) during editing.

Parameters –-> - a color box enabling setting the color of string (the left box) and background (the right one) after completion of editing, but before sending in mode with acknowledge.

Parameters – Window background - a check box that enables setting the background color the same as the window background color.

Parameters – Transparent background - a check box that enables setting a transparent backgroung.

Justify - To Left - a check box that forces displaying the text in the object box justified to left.

Justify - Center - a check box that forces centering the text in the object box.

Justify - To Right - a check box that forces displaying the text in the object box justified to right.

Configuring from the Variable Definitions Database

The STRING object has features, allowing setting the text variable length from the variables base. This object has ***Link*** button. Pressing this button causes inserting „@ number" characters into ***Length*** field, where number means the content of the field *ElementsCount* in the database. *Link* button works on condition, that the *Length* field is empty.

18.4.17. PICTURE Object

Dimensiosns
Object Parameters
Object Name
Bitmap name

The PICTURE object enables the bitmap to be displayed over the diagram. This is a static object.

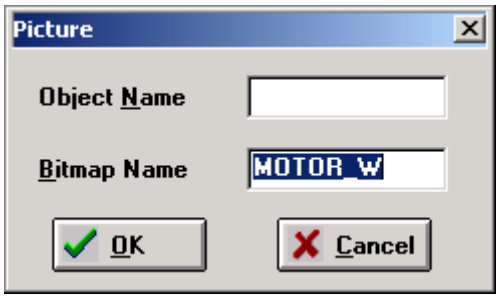


Figure 18-29. The PICTURE Example.

Dimensions

Dimensions of the PICTURE object are fixed (as determined on the basis of the bitmap size) and cannot be altered.

Parameters

Object Name - this text box is used to put into it the optional name of the object. If the name of the object will not be defined, its type together with coordinates will appear on the list of objects.

Bitmap name - a text field used for entering the bitmap's name. The bitmap's name can be entered in the dialog field or it can be chosen by clicking

the right mouse button. Then a window appears, enabling browsing and selection of bitmaps being part of **asix** system map pool.

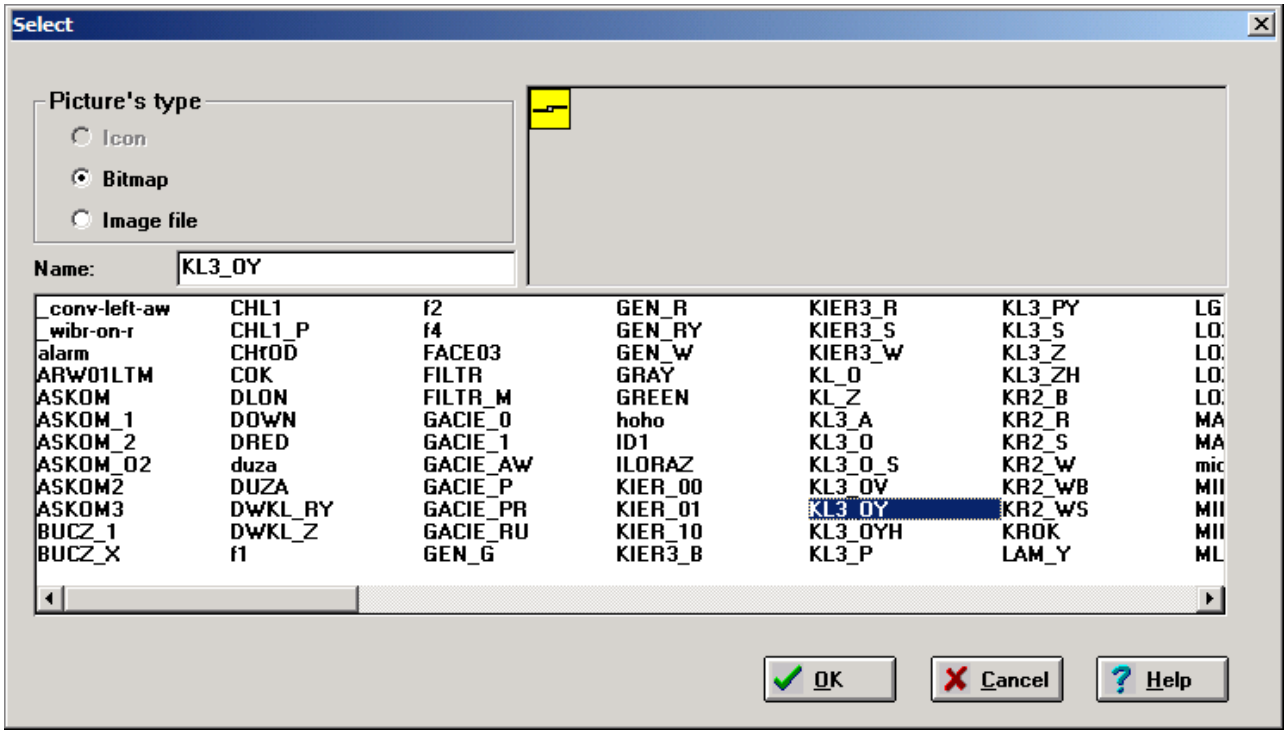


Figure 18-30. The PICTURE Object Selection Window.

The bitmaps may be created and imported as well by means of an **asix**-embedded bitmap editor. For accessing the bitmap editor use the TOOLS menu in the designer window.

18.4.18. PICTURES Object

Dimensions
Object Parameters
- Object Name
- Coding
- Bit Mask
- Picture List
- Sending
- State
- Variable

Coding Example

The PICTURES object enables one of many (up to 17) specified bitmaps to be displayed over the diagram. This is a dynamic object (if it is a multistate object). The picture selected for displaying depends on the value of the monitored variable. Optionally, the PICTURES object may also be selectable

and may serve to send a determined value to the controlled variable. Sending of value can be protected by a password.

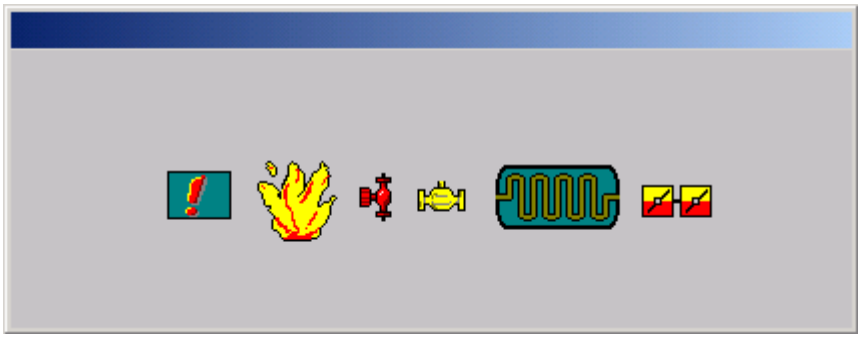


Figure 18-31. The PICTURES Example.

Dimensions

The dimensions of the PICTURES object are automatically mached to bitmap dimensions.

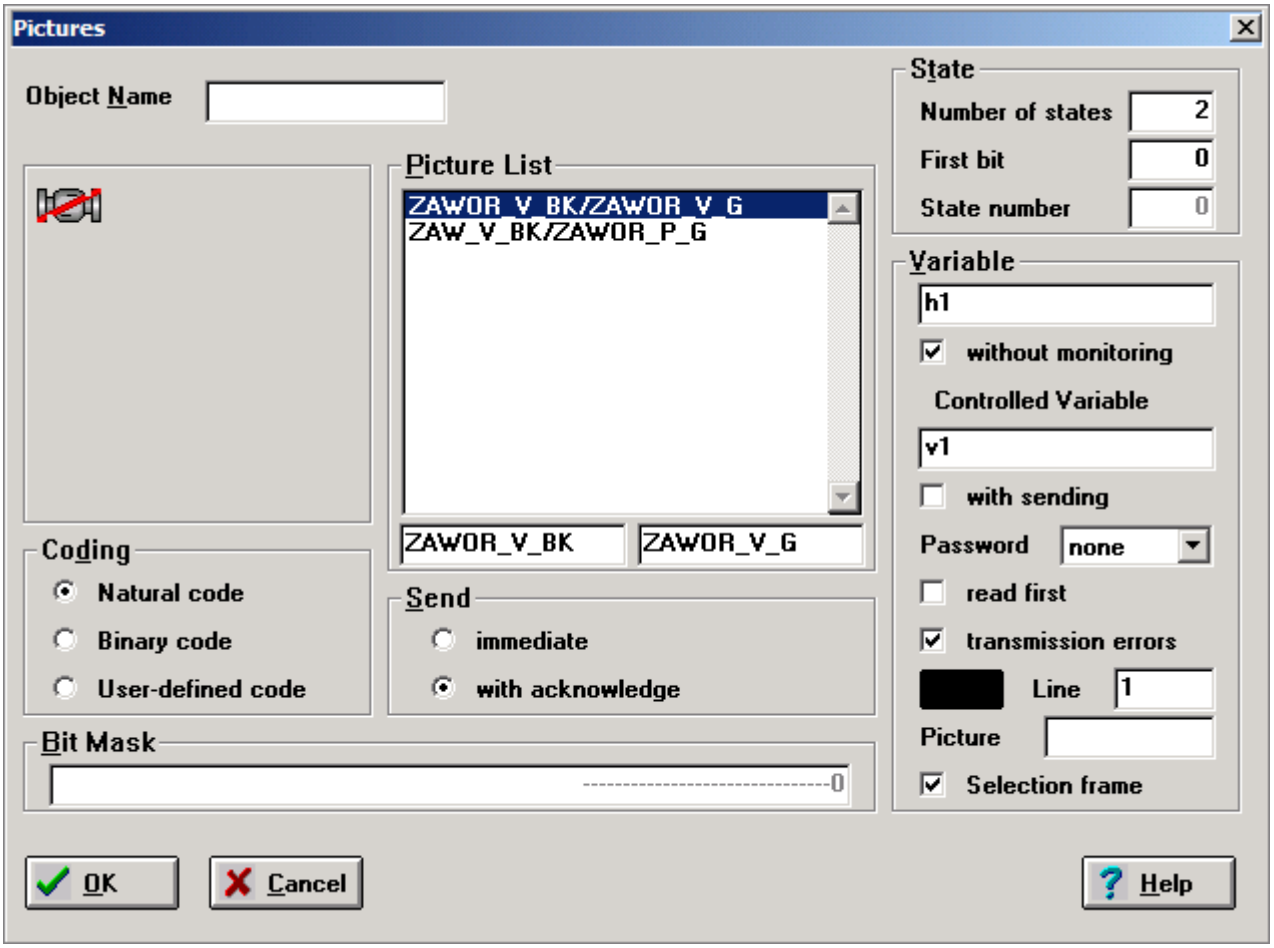


Figure 18-32. The PICTURES Object Parameterization Window.

The name of the bitmap can be either written in the dialog box or selected by clicking the right mouse button on the bitmap box. A window will appear to enable viewing and selecting the bitmaps which belong to the **asix** bitmap pool. The bitmap name is to be inserted to the list by pointing the proper place with the mouse cursor. Doing so, the bitmap name will be displayed in a special box of the dialog window (below the ***Object Name*** box).

Parameters

Object Name - this text box is used for to put in the optional name of the object. If the name of the object will not be defined, its type together with coordinates will appear on the list of objects.

Send Immediate - a radio button to declare immediate sending the value, after the proper bitmap is selected, by clicking the mouse button on the object box while being under the diagram refresh mode (only possible if the object has been set as controllable one). This type of sending is reasonable if the object has two states otherwise the object state previewing as resulted from the selection of the successive states will cause sending of a series of values. The value by object is the number that corresponds to the state number of object

Send with Acknowledge - a radio button to declare sending the value after the proper bitmap is selected by clicking (many times) the mouse button on the object box while being under the diagram refresh mode (only possible if the object has been set as controllable one) and then by acknowledging. Acknowledge of sending can be made by either clicking the button *Grey+* or executing appropriate actions. This type of sending is reasonable if the object is a multistate object. The value by object is the number that corresponds to the state number of object.

State - Number of States - a numeric box that enables declaring a number of states of the object. By default the object has 1 state and then it is the static object. Maximum number of states is 33 for natural coding, 64 for binary coding and user-defined coding. The individual states correspond with values that are result of the chosen coding mode.

State - First Bit - a numerical box used to declare the first (least significant) bit of the monitored variable the value of such variable is displayed as one of many declared pictures. The contents of this box (valid for both natural and binary coding only) permit for an unambiguous determination of the position of the bits group which control the pictures as displayed in the word. The first bit corresponds to the LSB whilst the group size corresponds to the coding method.

State - State Number - a numerical box only used to display the number of the state selected. Selection is made by clicking on the appropriate position of ***Picture List***.

Picture List - a list of names of the pictures (bitmaps) corresponding to the individual states of the object. To modify the picture name, choose a suitable position from the list, write the new name beneath the list and click *Enter*. To insert or select the bitmap name in the dialog box click the right mouse button. A window will appear to permit viewing and selecting the bitmaps belonging to the **asix** bitmap pool. Trying to declare the name of a non-existing picture will fail.

The *PICTURES* object enables declaring two bitmaps for each state. If the designer declares two bitmaps for each state, then the object will be alternately displaying both declared bitmaps. Names of these bitmaps are defined In two eddition fields placed below *Picture List*. Blinking option is activated when the name of bitmap will be passed in the left field. In *Picture List* the name of bitmaps are separated with '/' character for the states with blinking.

There is the possibility of declaring only one bitmap for the given state to obtain the blinking effect of single bitmap. In such case, one should pass the name consisting of characters '-' as the name of the alternative bitmap.

The frequency of blinking is constant and deifned in ***Blinking period*** parameter in the application configuration file (with use of Architect program: Architect > *Fields and Computers* > *Masks* module > *Misc* tab).

Coding - Natural Code - a radio button used to choose a natural code (for coding methods - described after PARAMETER list).

Coding - Binary Code - a radio button used to choose a binary code (for coding methods - described after PARAMETER list).

Coding - User-Defined Code - a selection box used to choose an user-defined code (for coding methods - described after PARAMETER list).

Variable - a text box used to specify the name of the monitored variable. Specifying the name is not necessary if the option ***Without monitoring*** has been selected. Clicking the right mouse button on this box opens a window containing the list of the available variables to make selection.

Control Variable - a text box used to specify the name of the control variable. Specifying the name is necessary if the option ***With sending*** has been selected. Clicking the right mouse button on this box opens a window containing the list of the available variables to make selection.

Without monitoring **-** a check box used to set the object parameters in such manner that the displayed state is not dependent on any system variable and serves only to indicate the value to be sent.

With sending **-** a check box used to set the object parameters in such manner that the value corresponding to its state could be sent to the system variable. Sending of the value requires the object to be selected, e.g. by clicking the mouse; what will be done afterwards depends on the sending mode.

Password - combo box of the password that protects sending the controls (four levels of password can be selected, no password is the default setting).

read first **-** a check box used to force reading before sending the controlled variable. In such case, there is written to the controlled variable a logic sum of the sent value and of the given variable value as read from the controller with the reset bit group which has been set by the object.

Transmission error **-** a check box used to warn about transmission errors (either the object is crossed or another bitmap is displayed instead of it). Default option.

Line - a color box of the crossing line signalling transmission error occurrence. A numerical box used to declare the line width. Selection of the color box causes displaying of the color choice window. See also description of the box ***Picture***.

Picture - a text box used to specify the bitmap (picture) name to be displayed when transmission errors occur. The name of the bitmap can be either written in the dialog box or selected by clicking the right mouse button on the bitmap box. A window will appear to enable viewing and selecting the bitmaps, which belong to the **asix** bitmap pool. If this box is empty, then on occurring of the transmission error the object is crossed only with the line defined by the ***Line*** box settings. If the box is filled, then a specified picture is displayed instead of the line. It is recommended, however, to use a picture for signalling transmission errors because of the crossing line remainders in the case of using transparent color to draw up the bitmaps.

Bit Mask - a numerical box to display the monitored variable (the variable, at the same time, which can be sent from the object if the parameters of this latter have been set as ***With sending***) which corresponds to the given state. Under user-coding, the mask can be set by inserting a hexadecimal or binary code corresponding to

the given state, possibly indicating the *don't care* bits with „-".

The ***'Coder'*** window is used to defie the states in user-defined coding mode:

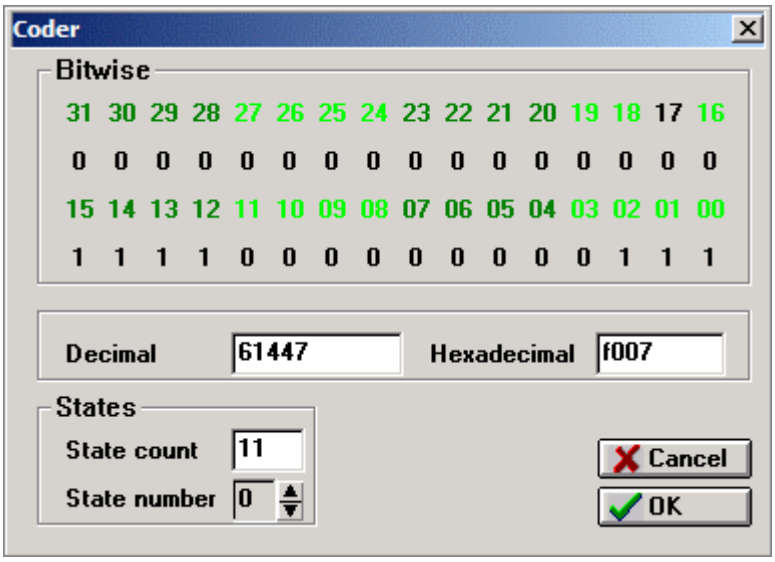


Figure 18-33. The 'Coder' Window.

The number of states recognized by the object is passed in *State count* field. This value may be also set in the main window for object defining. The state the code of which should be defined is selected in *State number* field. While setting the number in *Bitwise* frame, the current state coding is displayed - to change it, one should click with mouse on a bit value. Successive click operations cause change of the value in the following sequence:

- – bit value is unimportant
1 – bit has to be set
0 – bit has to be set to zero

Sate coding is automatically stored in the moment of choosing the other state.

The *Decimal* and *Hexadecimal* fields display decimal and hexadecimal value of defined state.

Selection Frame - a check box that permits additional displaying of the so called *Selection Frame* (Object contour) after the object is selected during the diagram refresh. The frame helps to know quickly which object is the last selected one.

During the refresh operation, the object can be selected using either the mouse cursor or the *Tab* button (*Shift+Tab*). After the selection is made, choice the given object state (which is signaled by a change of the picture) with the mouse cursor or the *Enter* key can by made. Sending of the state is done automatically after selection/acknowledge is made. To acknowledge, press the *Grey+ key* or carry out appropriate actions. If you don't want to do selection, click either *Esc* or the right mouse button.

There exist the possibility to use the conversion function, the values of it are the indexes of successive pictures as well as to use the another conversion function which changes the object state number to the sent value (when the option *With control* is active). The decoding of a fixed-point monitored variable value (and coding of possibly sent values as well) using one of three object-embedded strategies is possible too.

Natural Coding

The group of bits is checked. Number of bits is equal to the *number of states*, but number of the LSB defines "offset". The group of bits can consist only one value of "1". Number of the state is calculated as position of the "1" within the group of bits (the LSB position is number one). The zero state corresponds with the group of all zeros.

Binary Coding

The group of bits is checked. Number of bits is equal to the binary logarithm of *number of states*, but number of the LSB defines "offset". Number of state is calculated as content of the tested bit group.

User-defined ccoding

The whole word is checked. The "offset" parameter is ignored. The user assigns a string of ones and zeros (the *don't carry* states can be masked by "-") to each state. Number of state is calculated as the first of states, that the user assigned the same bits on the appropriate positions (regardless to the masked bits).

For the object, when parameter *with sending* has been set for it, the use of option *read first* is possible. In such a case the logical sum of the value being sent and the given value that has been read from the controller (the bits being controlled by the object are reset to zero) is written to the variable.

EXAMPLE

The object has four states and the natural coding has been chosen. To code four natural states (000,001,010,100) a three-bit group in the word is necessary. The first (smallest) bit of the group determines the contents of the box ***State-First bit*.** If it equals for example to 0, then the state of only three first bits affect the object state. The remaining bits has no influence on displaying of the object what is illustrated below.

----------xxx

("-" denotes a non-affecting bit, "x" denotes a bit which affects the object state)

If you define four pictures e.g. "1", "2", "3" and "4" (consecutively) which correspond to these four states, the picture „1" correspond to the state „000", the picture „2" correspond to the state „001" and so on. With writing, if you choose the picture „4", the corresponding state „100" of the smallest bits i.e. the decimal number 4 is sent to the variable. When the ***Read first*** box is activated, this mechanism is somewhat different - at first the given variable is

read and then the group of three smallest bits is reset, the state corresponding to the number of the pictures selected being overwritten.

```
1000000000100001    / the read variable
1000000000100---    / the read variable with reset bits „-"
------------100     / bits which are set by the object
1000000000100100    / the sent variable
```

18.4.19. PIPELINE Object

Dimensions
Object Parameterization
The list of parameters set for all the states
- Object Name
- Pipe Thickness
- Pattern
- Line
- Begin
- End

The list of parameters set for the individual states
- Variable
- State
- Bit Mask
- Line
- Coding

The PIPELINE object enables a pipeline portion to be displayed on the diagram. This is a dynamic object. It is possible to select the pipe thickness as well as to define individually each its constituting line, obtaining easily for example a three-dimensional effect. The pipeline ends may be declared separately to help drawing of elbows, tees etc. The pipeline portions can be drawn horizontally or vertically or inclined 45°. Besides when defining the proper line parameters for individual states of monitoring data it is possible to create the effect of animation.

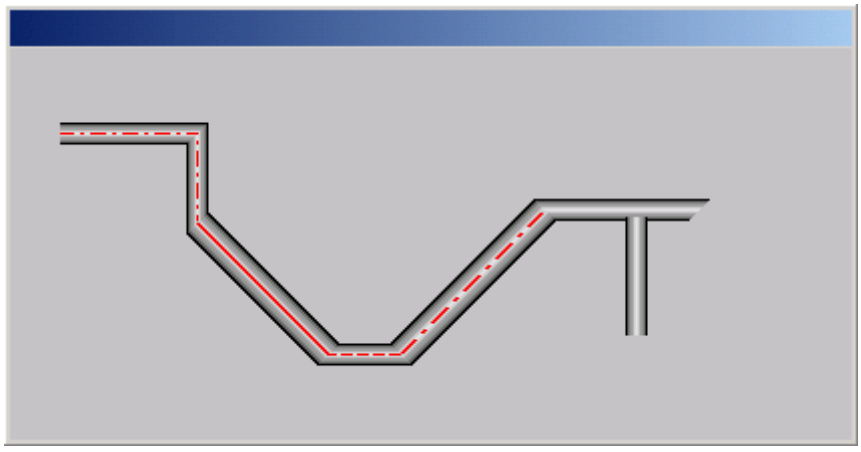


Figure 18-34. The PIPELINE Example.

Dimensions

One dimension of the PIPELINE object is fixed (as determined on the basis of the pipeline thickness) while the other is specified using the mouse (it is also possible to change the direction of PIPELINE by means of mouse).

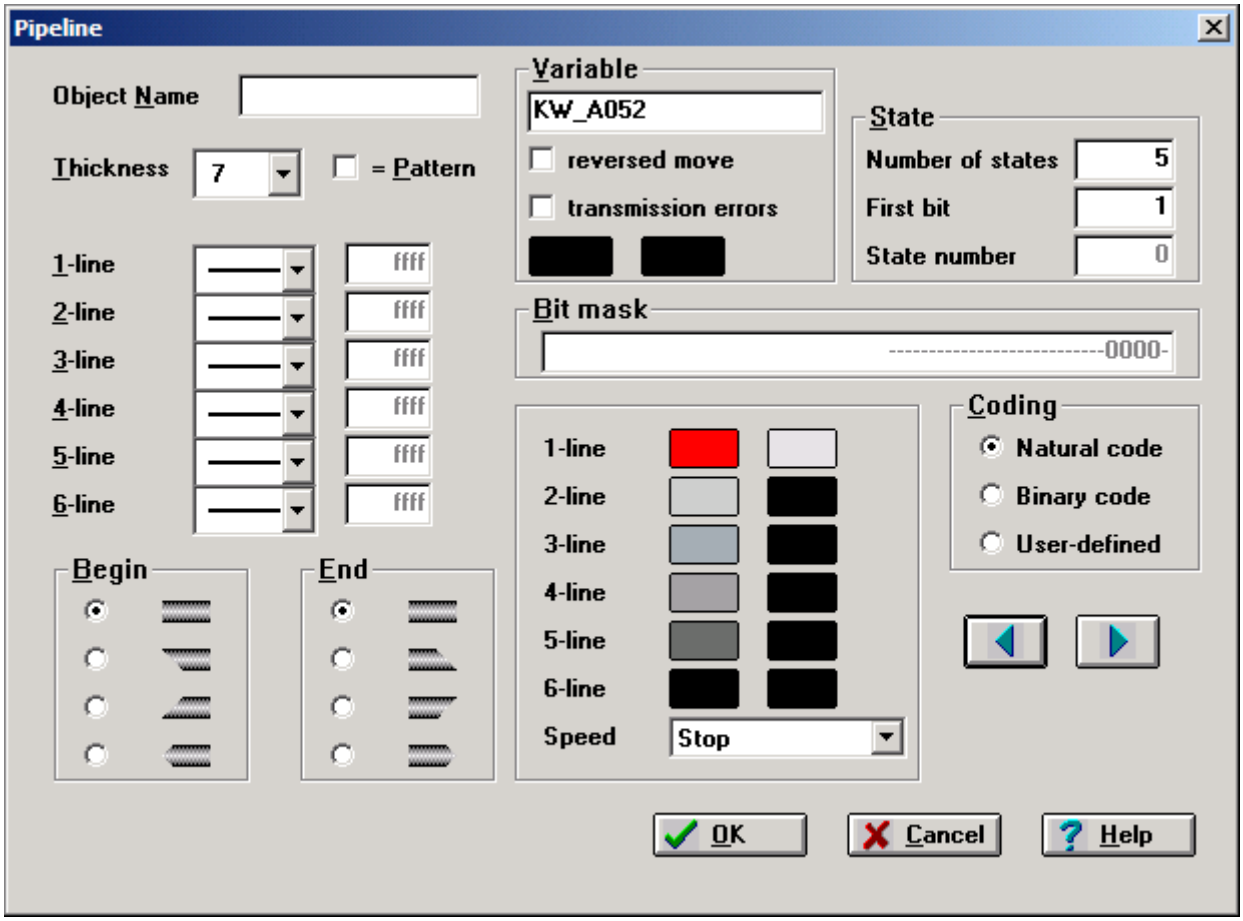


Figure 18-35. The PIPELINE Object Parameterization Window.

The pipeline is created from a line in such manner, that in the middle there is drawn a line denoted with number 1, being at the same time the symmetry axis for the successive lines to be drawn and denoted with 2, 3... and so on until the declared pipe thickness is obtained.

The box ***=Pattern*** determines the method of alignment of the line pattern with the pipeline begin. The pattern can be drawn in manner that the pipeline ends would be flat cut; the pattern might eventually „match" the shape of the pipe begin (either left or upper end depending on the orientation).

Parameters

The list of parameters set for all the states

Object Name - this text box is used to enter into it the optional name of the object. If the name of the

object will not be defined, its type together with coordinates will appear on the list of objects.

Pipe Thickness - a combo box provided to declare the pipe thickness in terms of pixels (only odd values of thickness are available).

= Pattern - a check box provided to switch the object on (justify). See description of the object).

i- Line - a set of boxes consisting of two color boxes, a combo box and a text box. These boxes serve to define individual lines, which create the pipeline. It is possible to define for each line the color, the second color (if this is a broken line), the line pattern (solid line, a variety of broken lines and an own pattern are available) and the hexadecimal value corresponding to the own pattern.

Begin - a radio button set provided to declare a pipe beginning portion as required (either left or upper - flat cut, left cut, right cut, sharpened).

End - a radio button set provided to declare a pipe ending portion as required (either right or lower - flat cut, left cut, right cut, sharpened).

The list of parameters set for the individual states

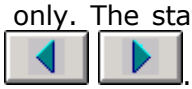
Variable - a text field that allows to pass a name of the monitored data; declaration is obligatory when the number of states is higher than 1; by pushing the right mouse button in this field it is possible to open the window with the disable variable list and to do selection.

Reversed Moved - this attribute allows to make the move of pipeline reversed (from right to left).

Transmission Errors - a check box provided to warn about transmission errors. When they occur, the object is crossed with the line of specified parameters. This is default option.

State-Number of states **-** a numeric box that enables declaring a number of states of the object. By default the object has 1 state and then it is a static object. Maximum number of states is 33 for natural coding, 64 for binary coding and user-defined coding. The individual states correspond with values that are result of the chosen coding mode.

State-First bit - a numerical box provided to declare the first (least significant) bit of the monitored variable, the value of such variable is displayed as one of many declared polygons. The contents of this box (valid for both natural and binary coding only) permit for an unambiguous determination of the position of the group of the bits, which control the states displayed in the word. The first bit corresponds to the LSB whilst the group size corresponds to the coding method.

State - State Number - a numerical box provided to display the number of the state selected only. The state selection is made by the buttons  .

Bit Mask - a numerical box provided to display the monitored variable (and the variable at the same time, which has been sent from the object being set as ***With control***) corresponding to the given state.

The ***'Coder'*** window is used to define the states in user-defined coding mode:

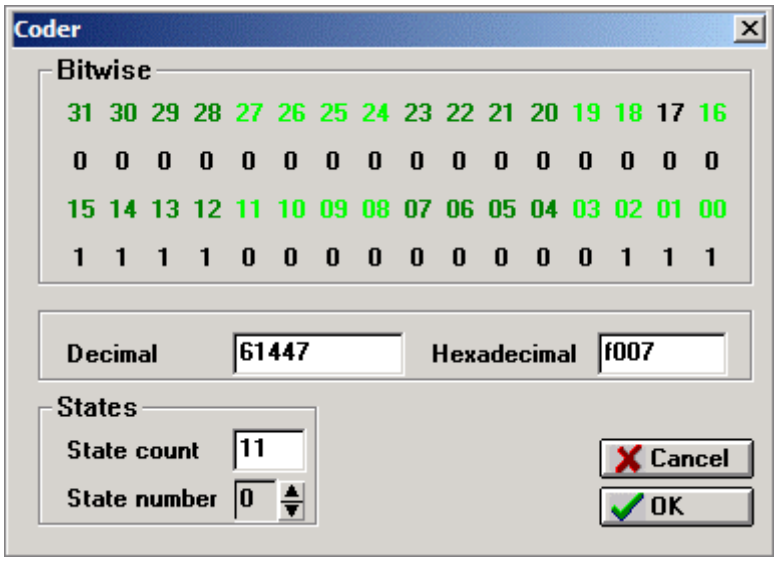


Figure 18-36. The 'Coder' Window.

The number of states recognized by the object is passed in *State count* field. This value may be also set in the main window for object defining. The state the code of which should be defined is selected in *State number* field. While setting the number in *Bitwise* frame, the current state coding is displayed - to change it, one should click with mouse on a bit value. Successive click operations cause change of the value in the following sequence:

- – bit value is unimportant
1 – bit has to be set
0 – bit has to be set to zero

Sate coding is automatically stored in the moment of choosing the other state.

The *Decimal* and *Hexadecimal* fields display decimal and hexadecimal value of defined state.

i-Line - a set of fields – each consisting of 2 color fields. These fields enable definition of particular lines creating a pipeline. It is possible to define a color and a second color (if the line is discontinuous).

Coding - Natural Code	- a radio button provided to choose a natural code (for coding methods - see chapter after Parameter List).
Coding - Binary Code	- a radio button provided to choose a binary code (for coding methods - see chapter after Parameter List).
Coding - User-Defined Code	- a radio button provided to choose an user-defined code (for coding methods - see chapter after Parameter List).

There is possibility to use the calculation function, the values of which are the indexes of successive polygons as well as to use the another recalculation function which changes the object state number to the sent value. The decoding of a fixed-point monitored variable value (and coding of possibly sent values as well) using one of three object-embedded strategies is possible too.

Natural Coding

The group of bits is checked. Number of bits is equal to the *number of states*, but number of the LSB defines "offset". The group of bits can consist only one value of "1". Number of the state is calculated as position of the "1" within the group of bits (the LSB position is number one). The zero state corresponds with the group of all zeros.

Binary Coding

The group of bits is checked. Number of bits is equal to the binary logarithm of *number of states*, but number of the LSB defines "offset". Number of state is calculated as content of the tested bit group.

User-Defined Coding

The whole word is checked. The "offset" parameter is ignored. The user assigns a string of ones and zeros (the bit can be masked by "-") to each state. Number of state is calculated as the first of states, that the user assigned the same bits on the appropriate positions (regardless to the bits outside the mask).

18.4.20. POINTER Object

Dimensions
Object Parameters
- Object Name
- Colors
- Scale
- Ranges
- Angles
- Frame
- Rotation
- Variable
- Outlook

Configuring from the VarDef

The POINTER object enables an analogue (circular-type) indicator to be displayed over the diagram. The pointer displays a system variable. This is a dynamic object, which is modified in function of its corresponding measurement variable. It is possible to change its color. The pointer may be provided with markers, which correspond to determined fixed or variable values. The pointing element can be either a classic hand (hands), an arc or a circular sector. The object is a selectable object if it is provided with two hands but it does not write any variables.

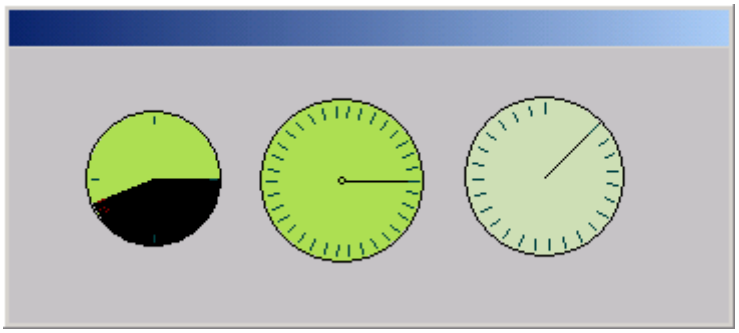


Figure 18-37. The PIONTER Examples.

Dimensions

The POINTER object dimensions are determined by means of the mouse. It cannot be an ellipse.

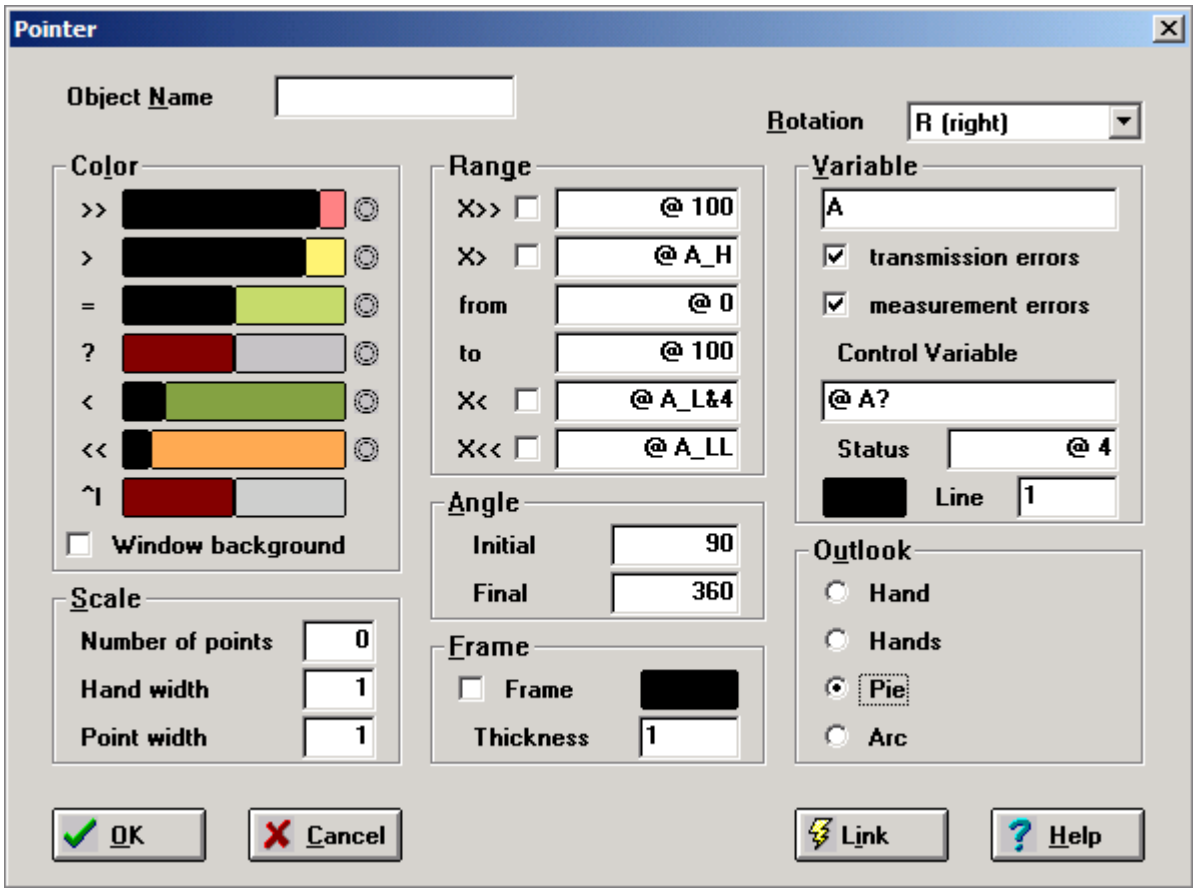


Figure 18-38. The PIONTER Object Parameterization Window.

Parameters

Object Name - this text box is designed to enter the optional name of the object. If the name of the object will not be defined, its type together with coordinates will appear on the list of objects.

Rotation - a combo box provided to determine the direction of the pointer rotation. The options are: R (right), L (left). The default option is R.

Color ->> - color fields enabling setting the indicator color (left field) and background color (right field) in case of exceeding the upper alarm limit. Indicator to the right from the color fields enables setting the blinking attribute.

Color -> - color fields that enable setting the indicator color (left field) and background color (right field) in case of exceeding the upper warning limit. Indicator to the right from the color fields enables setting the blinking attribute.

Color -= - color fields that enable setting the basic indicator (left field) and background color(right field) of displayed number. Indicator to the right from the color fields enables setting the blinking attribute.

Color -? - color fields that enable setting the indicator color (left field) and background color (right field) in case of error. Indicator to the right from the color fields enables setting the blinking attribute.

Color -< - color fields that enable setting the indicator color (left field) and background color (right field) in case of exceeding the lower warning limit. Indicator to the right from the color fields enables setting the blinking attribute.

Color -<< - color fields that enable enabling setting the indicator color (left field) and background color (right field) in case of exceeding the lower alarm limit. Indicator to the right from the color fields enables setting the blinking attribute.

Color -^| - color fields that enable enabling setting the scale color (right field) and markers color (left field).

Color - Window background - a check box provided to force the background color being the same as the Background window color.

Scale – Number of points **-** a numerical box provided to declare a scale consisting of the given number of markers. The default value is 0, what means "no scale".

Scale - Hand width **-** a numerical box provided to declare a hand width. The default value is 1.

Scale - Point width **-** a numerical box provided to declare a point width. The default value is 1.

Range -X>> - a check box and an alphanumeric box provided to specify either a numerical value of the upper alarm limit or a name of the variable, which contains this limit. The default value is 90. When the variable name is to be specified, its logic condition may be additionally specified in the box situated rightward. See the next sections for details on the conditions. The check box forces drawing of a marker.

Range -X> - a check box and an alphanumeric box to specify either a numerical value of the upper warning limit or a name of the variable which contains this limit. The default value is 75. When the variable name is to be specified, its logic condition may be additionally specified in the box situated rightward. See the next sections for details on the conditions. The check box forces drawing of a marker.

Range - from - a numerical-text box provided to specify either a numerical value or a name of the variable corresponding to the lower range of the pointer. The default value is 0.

Range - to - a numerical-text box provided to specify either a numerical value or a name of the variable corresponding to the upper range of the pointer. The default value is 100.

Range -X< - a check box and a numerical-text box provided to specify either a numerical value of the lower warning limit or a name of the variable, which contains this limit. The default value is 25. When the variable name is to be specified, its logic condition may be additionally specified in the box situated rightward. See the next sections for details on the conditions. The check box forces drawing of a marker.

Range -X<< - a check box and a numerical-text box to specify either a numerical value of the lower alarm limit or a name of the variable which contains this limit. The default value is 10. When the variable name is to be specified, its logic condition may be additionally specified in the box situated rightward. See the next sections for details on the conditions. The check box forces drawing of a marker.

Angle-Initial **-** a numerical box provided to specify the initial angle of the pointer range in terms of grades. The default value is 0.

Angle-Final **-** a numerical box provided to specify the final angle of the pointer range in terms of grades. The default value is 360.

Frame - Frame **-** a check box and a color box provided to specify whether the pointer should be provided with a frame and to specify the color of the frame.

Frame – Thickness **-** a numerical box to specify the frame thickness (Default 1).

Variable - a text box provided to specify a name of the measurement variable. Specifying the variable name is obligatory. By clicking the right mouse button on this box you can open a window containing the list of available variables to make selection.

Transmission Errors **-** a check box provided to warn about transmission errors. When they occur, the pointer is crossed with the line of specified parameters. This is default option.

Measurement Errors **-** a check box to warn about measurement errors. When they occur, the pointer is crossed with a circle (inside) of specified parameters. It requires specifying the ***Control Variable***.

Control Variable **-** a text box that allows specifying the name of control data. Specifying it is necessary, if the ***Measurements errors*** option has been selected. By pressing the right mouse button in this field, a window containing the list of available variables can be opened and you can make a choice in it. It is possible to specify the name of controlled variable using notation with **#** character. If in the field of variable name the #_ characters with suffix are entered, then during the application's run it will be treated as

	the variable with name containing the name of monitored variable instead of the # character.
Status	- a text box to insert a hexadecimal mask. If the product of this mask and the control variable is different from zero then a measurement error occurs.
Line	- a color box of the line, which crosses the object when errors occur and a numerical box which serve to determine the line thickness. Selection of the color box causes displaying of the color selection window.
Outlook-Hand	**-** a radio button provided to select a one-hand mode of operation.
Outlook-Hands	**-** a radio button provided to select a double-hand mode of operation (in such a case, the second hand plays a role of a maximum value memory hand; its position may be cancelled by the operator).
Outlook-Pie	**-** a radio button provided to select a mode of operation, which uses a piece of circle as an indicator.
Outlook-Arc	**-** a radio button provided to select a mode of operation, which uses an arc, which is elongated in function of the displayed value change.

NOTICE Empty texts and the texts consisted of white spaces are acceptable in limit fields and using them is treated as abandoning verification.

The displayed pointer may change its color (along with the background color) on exceeding the limits. Exceeding can be specified as either the numbers (fixed) or the system variable names. If the exceeding is specified as a system variable name, a logic condition to be met can be specified in the box close to the name (rightwards).

The logic condition is a sequence consisting of a „&" or „|" character followed by a hexadecimal number (condition). With the „&"character, condition is met if the logic product of the variable and the condition equals to the condition whilst with the „|"character, condition is met if the logic sum of the variable and the condition is different from zero. In other words, in the first case, to change the attribute it is necessary to have all the bits set in the given variable whilst in the second case at least one bit of the condition must be 1. When specifying the condition, the „|"character may be neglected.

Moreover, markers (dashes) may be inserted in the places corresponding to the limit exceeding. The markers are to be drawn after setting „yes" in the check box close to the box where the limit exceeding are written in. The pointer dial may be provided with a scale. In the case of two hands, the second hand serves to "store in" the maximum value indication. Making selection (with, for example, the mouse) causes interposing both the hands each other.

Configuring from the Variable Definitions Database

The POINTER object has been additionally provided with features enabling setting the following parameters from variables base:

- range of variation e.g. 0, 100 according to the name of displayed variable,
- limits; the color will be changed when they are exceeded, e.g. of **Limit**, **Limit&10** variables names, according to the name of displayed variable,
- name and diagram of control variable.

In this purpose the ***Link*** button is used, pressing of which causes inserting the contents of given field in the base, preceded with **@** character into all positions, that may potentially be set from the variables base.

In the example below clicking the Link button has caused inserting the contents of fields from the base for:

- critical maximum limit - 100 value;
- maximum limit - A_H;
- minimum limit - A_L;
- critical minimum limit - A_LL;
- name and mask of control data – A? with hexadecimal mask 4.

18.4.21. POLYGON and POLYGON HV Object

Dimensions
Parameters
Object Name
Line Color
Filling Color
Fill Inside
Window Background

The POLYGON and POLYGON HV objects enable static polygons to be displayed on a diagram. In the case of the POLYGON HV object, an open polygon being the boundary of a polygon may only consist of lines, which are parallel or perpendicular to each other.

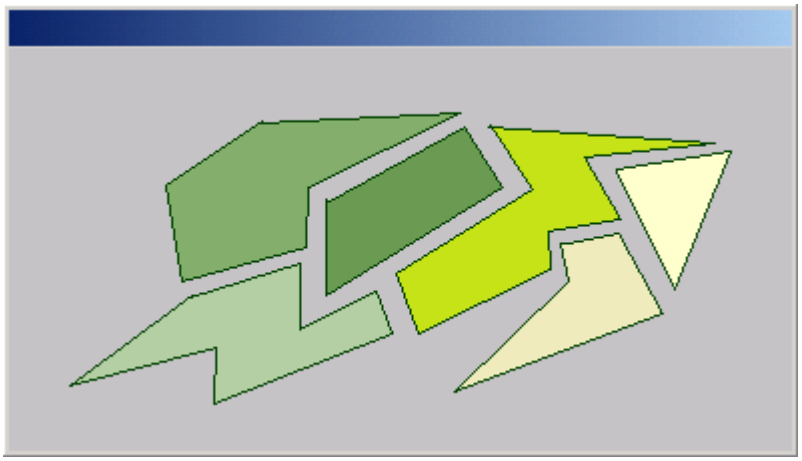


Figure 18-39. The POLYGON Examples.

Dimensions

The coordinates of the vertexes are defined with use of the mouse. The polygon creating line is ended by clicking with the right mouse button.

POLYGON- type objects can be rotated vertically/horizontally using appropriate edit commands.

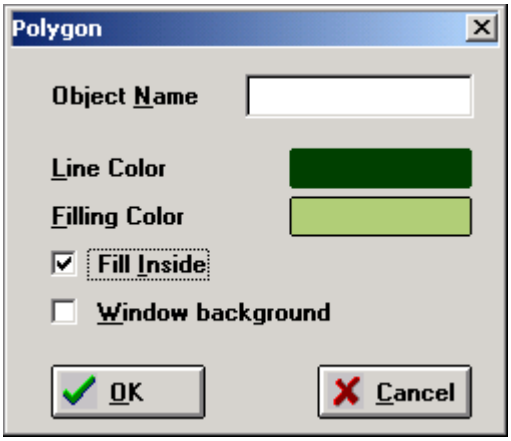


Figure 18-40. The POLYGON Object Parameterization Window.

Parameters

Object Name - this text box is designed to enter the optional name of the object. If the name of the object is not be defined, its type together with coordinates will appear on the list of objects.

Line Color - a color box provided to set color of the polygon creating line after the color selection window is activated.

Filling Color	**-** a color box provided to set color of the polygon inside after the color selection window is activated.
Fill Inside	**-** a check box provided to fill the polygon inside.
Window Background	- a check box provided to force drawing the polygon inside with the diagram background color.

18.4.22. POLYGONS and POLYGONS HV Objects

Dimensions
Object Parameters
- Object Name
- Parameters
- Sending
- State
- Line
- Background
- Blinking
- Coding
- Variable
- Bit mask

Example of Coding

The POLYGONS and POLYGONS HV objects enable polygons to be displayed on the diagram. In the case of the POLYGONS HV object, an open polygon being the boundary of a polygon may only consist of lines, which are parallel or perpendicular each other. The object as displayed may change its color in function of the monitored variable. This is a dynamic object (if it is a multistate object). Optionally, the POLYGONS object may be also a selectable object and may be used to send a specified value to the controlled variable. Sending of the value can be password-protected.

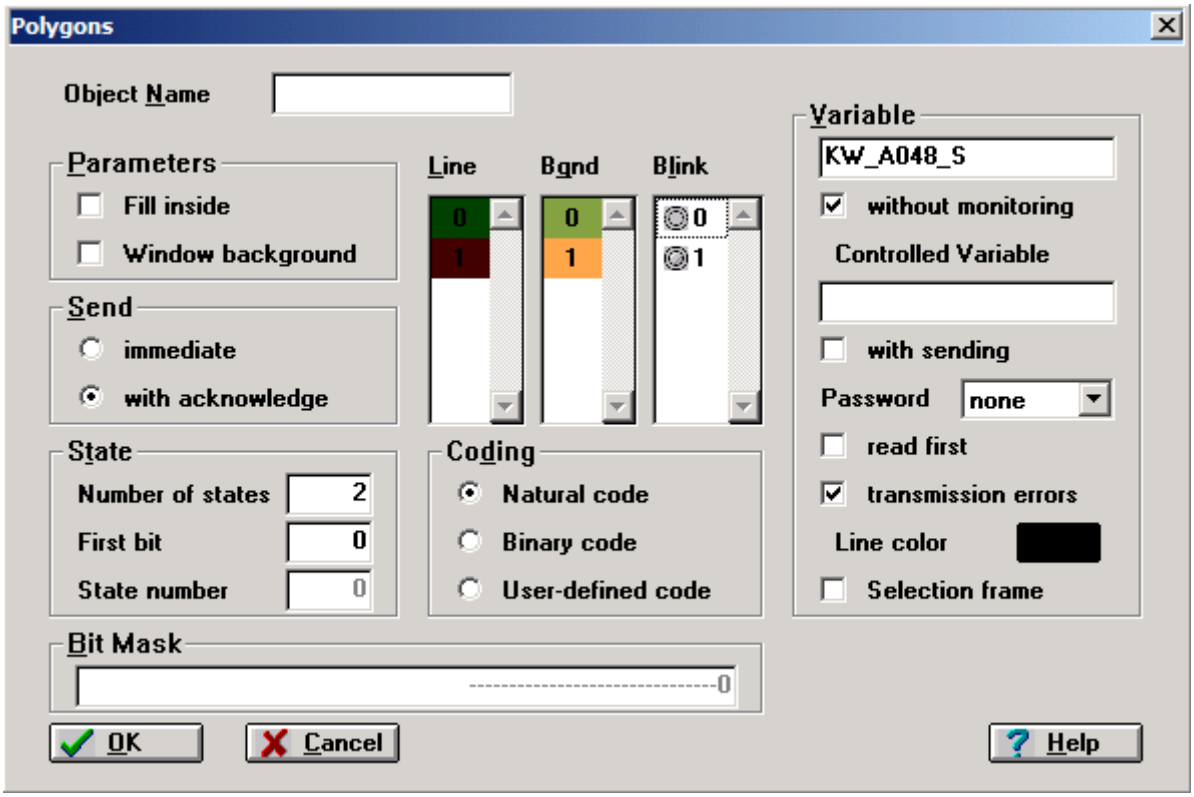


Figure 18-41. The POLYGONS Object Parameterization Window.

Dimensions

The coordinates of both the beginning and the end of the segments are defined with use of the mouse. The polygon creating line is ended by clicking with the right mouse button.

By default, the object has one state only i.e. it is a static object. Crossing the object with oblique line of specified color and thickness signals transmission errors.

During the refreshing operation, the object can be selected using either the mouse cursor or the *Tab* key (*Shift+Tab*). After the selection is made, choose the given object state (color) with the mouse cursor or the *Enter* button. Sending the state is done automatically after selection or when acknowledgement is made. To acknowledge, press the *Grey+* key or carry out appropriate actions. To quit the selection, click either *Esc* or the right mouse button.

POLYGONS - type objects can be rotated vertically/horizontally using appropriate edit commands.

Parameters

Object Name - this text box is designed to enter the optional name of the object. If the name of the object is

not be defined, its type together with coordinates will appear on the list of objects.

Parameters-Fill Inside - a check box provided to fill the polygon inside.

Parameters-Window Background - a check box provided to force drawing the polygon inside with the diagram background color.

Send Immediate - a radio button provided to declare immediate sending the value, after the corresponding polygon is selected, by clicking the mouse button on the object box while being under the diagram refreshing mode (only possible if the object is set ***With control***). This type of sending is reasonable if the object has two states otherwise the object state previewing as resulted from the selection of the successive states will cause sending of a series of values. The value by object is the number that corresponds to the state number of object.

Send with Acknowledge- a radio button provided to declare sending of the value after the corresponding polygon is selected by clicking (many times) the mouse button on the object box while being under the diagram refresh mode (only possible if the object is set ***With control***) and then by acknowledging. Acknowledge of sending can be made by either clicking the button "*Grey* +" or executing appropriate actions. This type of sending is reasonable if the object is a multistate object. The value by object is the number that corresponds to the state number of object.

State - Number of States - a numeric box that enables declaring a number of states of the object. By default the object has 1 state and then it is the static object. Maximum number of states is 33 for natural coding, 64 for binary coding and user-defined coding. The individual states correspond with values that are result of the chosen coding mode.

State - First Bit - a numerical box provided to declare the first (least significant) bit of the monitored variable, the value of such variable is displayed as one of many declared polygons. The contents of this box (valid for both natural and binary coding only) permit for an unambiguous determination of the position of the group of the bits, which control the polygons as, displayed in the word. The first bit corresponds to the LSB whilst the group size corresponds to the coding method.

State - State Number - a numerical box provided to display the number of the state selected only. Clicking on the appropriate item of either ***Color List*** makes selection.

Line - a color list corresponding to the individual lines, which create the polygon border. To

change color, select the appropriate position in the list and strike *Enter*. The color choice window will appear. Trying to declare color for a non-existing state will fail. The numbers of states (from 0 to n) are displayed black in the color list.

Background - a color list corresponding to the individual polygon insides. To change color, select the appropriate position in the list and strike *Enter*. The color choice window will appear. Trying to declare color for a non-existing state will fail. The numbers of states (from 0 to n) are displayed black in the color list.

Blinking - a list of blinking indicators that opens blinking attribute setting window for each of the statuses.

Coding - Natural Code - a radio button provided to choose a natural code (for coding methods - see chapter after Parameter List).

Coding - Binary Code - a radio button provided to choose a binary code (for coding methods - see chapter after Parameter List).

Coding - User-Defined Code - a radio button provided to choose an user-defined code (for coding methods - see chapter after Parameter List).

Variable - a text box provided to specify the name of the monitored variable. Specifying the name is not obligatory if the ***Without monitoring*** option has been selected. Clicking the right mouse button on this box opens a window containing the list of the available variables to make selection.

Controlled Variable - a text field that allows setting the name of controlled variable. Specifying it is necessary, if the ***With control*** option has been selected. By clicking the right mouse button in this field, a window containing the list of available variables can be opened and you can make a choice in it. It is possible to specify the name of controlled variable using notation with **#** character. If in the field of variable name the #_ characters with suffix are entered, then during the application's run it will be treated as the variable with name containing the name of monitored variable instead of the # character.

Without Monitoring **-** a check box to set the object parameters in such manner that the displayed state would be independent on any system variable but serves only to indicate the value to be sent.

With Sending **-** a check box to set the object parameters in such manner that the value corresponding to its state would be sent to the system variable. Sending of the value takes place after the object is selected, for example by clicking on it with the mouse - the action, which will follow depends on the send mode.

Password - a password combo box that protects sending of the controls (four levels of password can be selected, no password is the default setting).

Read First - a check box provided to force reading before sending of the controlled variable. In such case, there is written to the controlled variable a logic sum of the sent value and of the given variable value as read from the controller with the reset bit group which has been set by the object.

Transmission Errors - a check box provided to warn about transmission errors (the object is crossed). Default option.

Line Color - a color box provided to specify color of the line, which crosses the object when transmission errors occur. Selection of this box causes displaying of the color selection window.

Bit Mask - a numerical box provided to display the monitored variable (and the variable at the same time, which has been sent from the object being set as ***With control***) corresponding to the given state. Under user-defined coding, it is possible to set the mask by writing either hexadecimal or binary code corresponding to the given state with possibly marking of the don't care bits with the „-" symbol.

The ***'Coder'*** window is used to defie the states in user-defined coding mode:

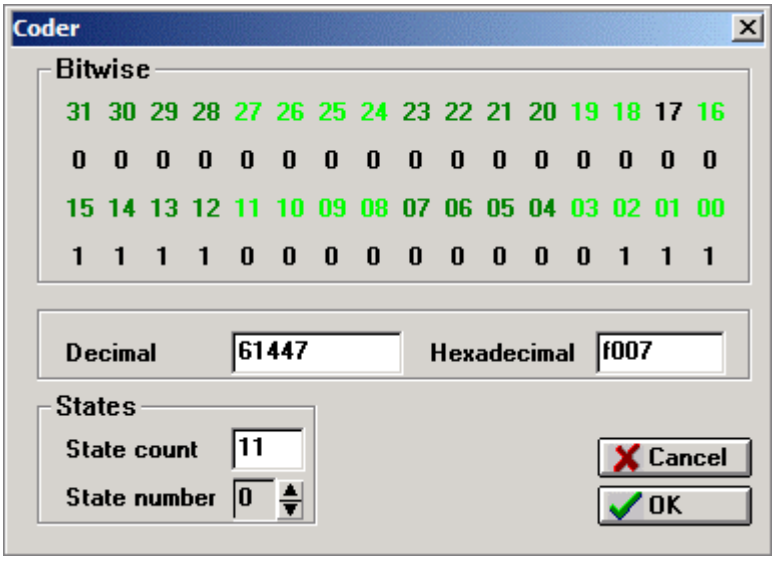


Figure 18-42. The 'Coder' Window.

The number of states recognized by the object is passed in *State count* field. This value may be also set in the main window for object defining. The state the code of which should be defined is selected in *State number* field. While setting the number in *Bitwise* frame, the

current state coding is displayed - to change it, one should click with mouse on a bit value. Successive click operations cause change of the value in the following sequence:

- – bit value is unimportant
1 – bit has to be set
0 – bit has to be set to zero

Sate coding is automatically stored in the moment of choosing the other state.

The *Decimal* and *Hexadecimal* fields display decimal and hexadecimal value of defined state.

Selection Frame - a check box that permits additional displaying the so called *Selection Frame* (object contour) after the object is selected during the diagram refreshing. The frame helps to know quickly, which object was the last selected one.

By default, the object has one state only i.e. it is a static object. Transmission errors are signaled by crossing the object.

During the refreshing operation, the object can be selected using either the mouse cursor or the *Tab* button (*Shift*+*Tab*). After the selection is made, choose the given object state (which is signalled by a color of the border and of the inside) with the mouse cursor or the *Enter* key. Sending the state is done automatically after selection/acknowledge is made. To acknowledge, press the *Grey*+ key or carry out appropriate actions. To quit the selection, click either *Esc* or the right mouse button.

There is possibility to use the calculation function, the values of which are the indexes of successive polygons as well as to use the another recalculation function which changes the object state number to the sent value (when the option *With control* is active). The decoding of a fixed-point monitored variable value (and coding of possibly sent values as well) using one of three object-embedded strategies is possible too.

Natural Coding

The group of bits is checked. Number of bits is equal to the *number of states*, but number of the LSB defines "offset". The group of bits can consist only one value of "1". Number of the state is calculated as position of the "1" within the group of bits (the LSB position is number one). The zero state corresponds with the group of all zeros.

Binary Coding

The group of bits is checked. Number of bits is equal to the binary logarithm of *number of states*, but number of the LSB defines "offset". Number of state is calculated as content of the tested bit group.

User-Defined Coding

The whole word is checked. The "offset" parameter is ignored. The user assigns a string of ones and zeros (the *don't carry* states can be masked by "-") to each state. Number of state is calculated as the first of states,

that the user assigned the same bits on the appropriate positions (regardless to the masked bits).

For the objects which are set *With control*, the option *Read first* may be used. In such case, there is written to the controlled variable a logic sum of the sent value and of the given variable value as read from the controller with the reset bit group which has been set by the object.

EXAMPLE

The object has four states and the binary coding has been chosen. To code four binary states (00,01,10,11) a two-bit group in the word is necessary. The first bit of the group determines the contents of the field ***State-First bit***. If it equals for example to 4, then the state of only the fifth and sixth bits affect the object state. The remaining bits have no influence on displaying of the object what is illustrated below.

----------xx----

(„-"denotes a non-affecting bit, "x" denotes a bit which affects the object state)

If you define four polygons e.g. „yellow starlet", „green starlet, „blue starlet" and „red starlet" (consecutively) which correspond to these four states, the „yellow starlet " corresponds to the state „00", the „green starlet " corresponds to the state „01" and so on. With writing, if you choose the „red starlet", the corresponding state „11" of the fifth and sixth bits i.e. the decimal number 48 is sent to the variable. When the ***Read first*** box is activated, this mechanism is somewhat different - at first the given variable is read and then the fifth and sixth bits are reset, the state corresponding to the color of the line selected being overwritten to these bits.

```
1000000000100000    / the read variable
1000000000—0000     / the read variable with reset bits „-"
----------11----    / bits which are set by the object
1000000000110000    / the sent variable
```

18.4.23. POLYLINE and POLYLINE HV Objects

Dimensions
Object Parameters
- Object Name
- Line Thickness
- Type
- Line Color
- Break Color
- Break
- Pattern
- Definition
- Round Line Edge

The POLYLINE and POLYLINE HV objects enable a sequence of line segments to be displayed over a diagram. In the case of the POLYLINE HV object, segments

can be vertical or horizontal. This is a static object. It is possible to specify the line color and thickness.

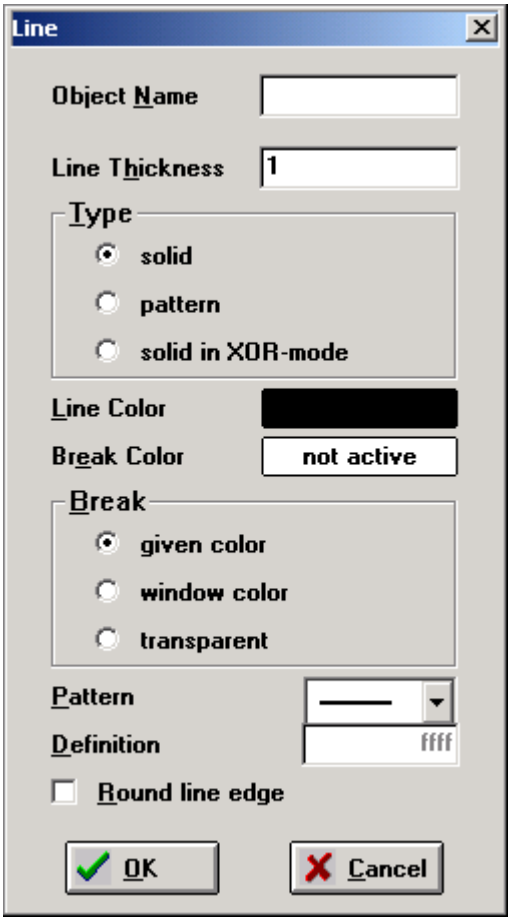


Figure 18-43. The POLYLINE Object Parameterization Window.

Dimensions

The coordinates of both the line beginning and end are determined with the mouse. The multi-line is ended by clicking with the right mouse button.

Parameters

Object Name - this text box is designed to enter the optional name of the object. If the name of the object is not be defined, its type together with coordinates will appear on the list of objects.

Line Thickness - a numerical box provided to specify the line thickness in terms of pixels. In the case of broken (e.g. dotted) lines, only odd values of thickness are admissible.

Type-Solid - a radio button provided to select a solid line.

Type-Pattern - a radio button provided to select a broken line.

Type-Solid in XOR-mode - a radio button to select a solid line but drawn in a XOR-mode, (i.e. such line, the color of which results as an XOR operation with the background color).

Line Color - a color box provided to set the color of the line. After selection, the color selection window appears.

Break Color - a color box provided to set the color of the line break (if it is the case of a broken line otherwise this box will be inactive). After selection, the color selection window appears.

Break - Given Color - a radio button provided to choose drawing of a broken line with the given color.

Break - Window Color - a radio button provided to choose drawing of a broken line with the window color.

Break - Transparent - a radio button provided to choose drawing of a transparent broken line.

Pattern - a combo box provided to select a pattern for a broken line (available there are a few patterns and an own pattern as well which is declared in the successive box). By default, it is a solid line.

Definition - a text box provided to select an own pattern or to display a standard pattern. The pattern is displayed as a hexadecimal number, the bits of which correspond to the successive points of the line. (For a solid line, for example there is displayed a pattern „ffff"). With an own pattern it is possible to edit this box (using the ***Pattern*** box).

Round Line Edge - a check box provided to round line edges. (Valid for broke lines only).

The POLYLINE objects can be rotated vertically and horizontally using appropriate edit commands. The color of the line drawn under XOR mode is determined as a difference between the declared line color and the background color. Thus, the line is always visible and changes its color automatically when being overimposed on various background colors.

18.4.24. POLYLINES and POLYLINES HV Object

Dimensions
Object Parameters
- Object Name
- Parameters
- Sending
- State
- Line
- Blinking
- Coding
- Variable
- Bit Mask

Exmaple of Coding

The POLYLINES and POLYLINES HV objects enable broken lines to be displayed on a diagram. In the case of the POLYLINES HV object, the broken line must consist of segments being vertical or parallel. This is a dynamic object (if it is a multistate object). Optionally, the POLYLINES object may be also a selectable object and may serve to send a specified value to the controlled variable. Sending of the value can be password-protected.

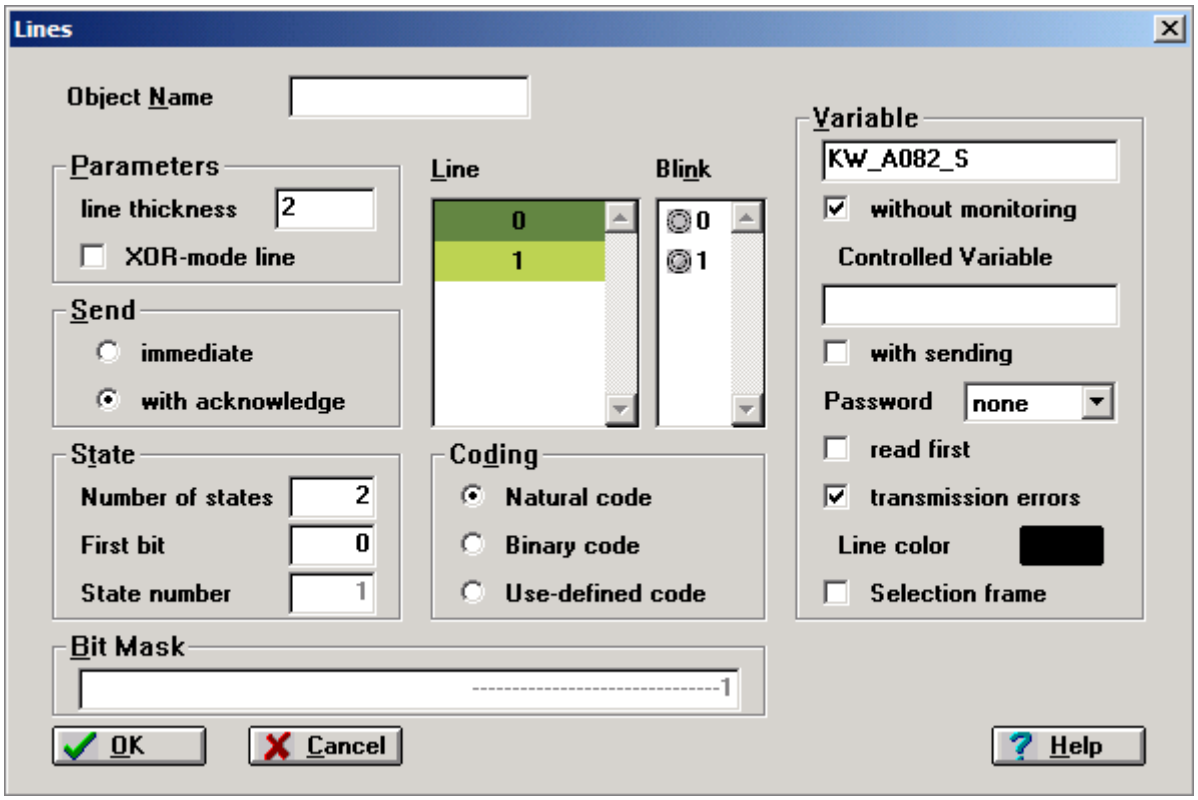


Figure 18-44. The POLYLINES Object Parameterization Window.

Dimensions

The coordinates of both the beginning and the end of the segments are specified with the mouse. The polyline is ended with the right mouse button.

By default, the object has one state only i.e. it is a static object. Crossing the line with a specified color signals transmission errors.

POLYLINES - type objects can be rotated vertically/horizontally using appropriate edit commands.

Parameters

Object Name - this text box is designed to enter the optional name of the object. If the name of the object will not be defined, its type together with coordinates will appear on the list of objects.

Parameters-Line Thickness - a numerical box provided to specify the line thickness.

Parameters - XOR-mode line - a check box provided to force drawing a line in a XOR-mode. The color of a line drawn under XOR is established as a symmetric difference between the declared colors of the line and of the background. Thus, the line is always visible and changes automatically its color when superimposed on different background colors.

Send - Immediate - a radio button provided to declare immediate sending of the value, after the corresponding multiline is selected, by clicking the mouse button on the object box while being under the diagram refresh mode (only possible if the object is set ***With control***). This type of sending is reasonable if the object has two states otherwise the object state previewing as resulted from the selection of the successive states will cause sending of a series of values.

Send - with Acknowledge - a radio button to declare sending of the value after the corresponding multiline is selected by clicking (many times) the mouse button on the object box while being under the diagram refresh mode (only possible if the object is set ***With control***). and then by acknowledging. Acknowledge of sending can be made by either clicking the button "*Grey+*" or executing appropriate actions. This type of sending is reasonable if the object is a multistate object.

State - Number of States - a numeric box that enables declaring a number of states of the object. By default the object has 1 state and then it is the static object. Maximum number of states is 33 for natural coding, 64 for binary coding and user-defined coding. The individual states correspond with values that are result of the chosen coding mode.

State - First Bit - a numerical box provided to declare the first (least significant) bit of the monitored variable, the value of such variable is displayed as one of many declared multi-lines. The contents of this box (valid for both natural and binary

coding only) permit for an unambiguous determination of the position of the group of the bits, which control the polygons as displayed in the word. The first bit corresponds to the LSB whilst the group size corresponds to the coding method.

State - State Number - a numerical box provided to display only the number of the state selected. Selection is made by clicking on the appropriate item of either ***Color List***.

Line Color - a color box to specify color of the line, which crosses the object when transmission errors occur. Selection of this box causes displaying of the color selection window.

Blinking - a list of blinking indicators opening blinking attribute setting window for each of the statuses.

Coding - Natural Code - a radio button provided to choose a natural code (for coding methods - see Parameter List).

Coding - Binary Code - a radio button provided to choose a binary code (for coding methods - see Parameter List).

Coding - Use-Defined Code - a radio button provided to choose an user-defined code (for coding methods - see Parameter List).

Variable - a text box provided to specify the name of the monitored variable. Specifying the name is not obligatory if the ***Without monitoring*** option has been selected. Clicking the right mouse button on this box opens a window containing the list of the available variables to make selection.

Controlled Variable - a text box that allows specifying the name of controlled data. Specifying this name is obligatory, if the ***With control*** option has been selected. By pressing the right mouse button in this field, a window containing the list of available variables can be opened and you can make a choice in it. It is possible to specify the name of controlled variable using notation with **#** character. If in the field of variable name the #_ characters with suffix are entered, then during the application's run it will be treated as the variable with name containing the name of monitored variable instead of the # character.

Without Monitoring **-** a check box provided to set the object parameters in such manner that the displayed state would be independent on any system variable but serve only to indicate the value to be sent.

With Sending **-** a check box provided to set the object parameters in such manner that the value corresponding to its state would be sent to the system variable. Sending of the value takes place after the object is selected, for example

by clicking on it with the mouse - the action, which follows will depend on the sending mode.

Password - a password combo box that protects against sending the controls (four levels of password can be selected, no password is the default setting).

Read First **-** a check box provided to force reading before sending of the controlled variable. In such case, there is written to the controlled variable a logic sum of the sent value and of the given variable value as read from the controller with the reset bit group which has been set by the object.

Transmission Errors **-** a check box provided to warn about transmission errors (the object is crossed). Default option.

Line Color - a color box provided to specify color of the line, which crosses the object when transmission errors occur. Selection of this box causes displaying of the color selection window.

Bit Mask - a numerical box provided to display the monitored variable (and the variable at the same time, which has been sent from the object being set as ***With control***) corresponding to the given state. Under user-defined coding, it is possible to set the mask by writing either hexadecimal or binary code corresponding to the given state with possibly marking of the bits such coded with the „-" symbol.

The ***'Coder'*** window is used to define the states in user-defined coding mode:

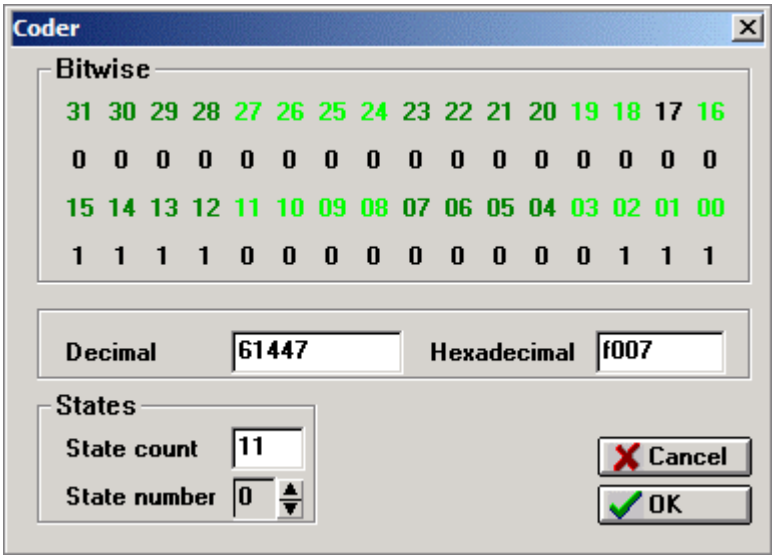


Figure 18-45. The 'Coder' Window.

The number of states recognized by the object is passed in *State count* field. This value may be also set in the main window for object defining. The state the code of which should be defined is selected in *State number* field. While setting the number in *Bitwise* frame, the current state coding is displayed - to change it, one should click with mouse on a bit value. Successive click operations cause change of the value in the following sequence:

- – bit value is unimportant
1 – bit has to be set
0 – bit has to be set to zero

Sate coding is automatically stored in the moment of choosing the other state.

The *Decimal* and *Hexadecimal* fields display decimal and hexadecimal value of defined state.

Selection Frame - a check box that permits additional displaying of the so called *Selection Frame* (object contour) after the object is selected during the diagram refresh. The frame helps to know quickly which object is the last selected one.

By default, the object has one state only i.e. it is a static object. Transmission errors are signalled by change of object color.

During the refresh operation, the object can be selected using either the mouse cursor or the *Tab* button (*Shift*+*Tab*). After the selection is made, choice the given object state (which is signalled by a color of the border end of the inside) with the mouse cursor or the *Enter* button. Sending of the state is done automatically after selection/acknowledge is made. To acknowledge, press the button *Grey+* or carry out appropriate actions. To quit the selection, click either *Esc* or the right mouse button.

There exist the possibility to use the conversion function, the values of it are the indexes of successive polyline colors as well as to use the another conversion function which changes the object state number to the sent value (when the option *With control* is active). Possible also is decoding of a fixed-point monitored variable value (and coding of possibly sent values as well) using one of three object-embedded strategies.

Natural Coding

A group of bits is verified. The number of bits is equal to the *number of states* whilst the number of the least significant bit determines the „shift". The bit group may comprise one 1 only. The state number is determined as the position of the 1 within the bit group (the smallest position has the number one). The null state corresponds to the group of zeros only.

Binary Coding

A group of bits is verified. The number of bits is equal to a $\log_2$ of the *number of states* whilst the number of the least significant bit determines the „shift". The bit group may comprise one 1 only. The state number is determined as the contents of the tested bit group.

User-Defined Coding

The entire word is verified. The „shift" parameter is of no concern. A sequence of 0 and 1 is assigned to each state by the user (with possibility of masking the bit with the character '-'). The state number is determined as the first state with the same bits assigned by the user at appropriate positions (neglecting the masked bits).

For the objects which are set *With control*, the option *Read first* may be used. In such case, there is written to the controlled variable a logic sum of the sent value and of the given variable value as read from the controller with the reset bit group which has been set by the object.

EXAMPLE

The object has four states and the binary coding has been chosen. To code four binary states (00,01,10,11) a two-bit group in the word is necessary. The first (smallest) bit of the group determines the contents of the field ***State-First bit***. If it equals for example to 4, then the state of only the fifth and sixth bits will affect the object state. The remaining bits have no influence on displaying of the object what is illustrated below.

----------xx----

(„-"denotes a non-affecting bit, "x" denotes a bit which affects the object state)

If you define four multi-lines e.g. „yellow", „green" , „blue" and „red " (consecutively) which correspond to these four states, the „yellow multiline" correspond to the state „00", the „green multiline" correspond to the state „01" and so on. With writing, if you choose the „red multiline", the corresponding state „11" of the fifth and sixth bits i.e. the decimal number 48 is be sent to the variable. When the ***First read*** box is activated, this mechanism is somewhat different - at first the given variable is read and then the fifth and sixth bits are reset, the state corresponding to the color of the line selected being overwritten to these bits.

```
1000000000100000    / the read variable
1000000000--0000    / the read variable with reset bits „-"
----------11----    / bits which are set by the object
1000000000110000    / the sent variable
```

18.4.25. PRESENTER Object

Dimensions
Object Parameters
Object Name
Bit Map
Opening Mode
Refresh Objects
Diagram Name

The PRESENTER object serves to point on the object selected (referred further as **pointed object**) provided that a certain condition is met. The condition for the PRESENTER object is the fact of displaying a specified diagram. Practically, the necessity of the use of this object arises when there is a large amount of controlled objects (for example drives or valves) located on one diagram. Normally, the problem of sending of controlled variables is approached by superimposing over the plant state object a transparent BUTTON-type object performing the action of displaying of a certain specialized diagram called "control panel" from which it is possible to send the controlled variables. If there are many such plants on the diagram then it is difficult to know at a glance which plant is associated to the "control panel". Difficulties are normally hard because the „control panels" are very similar each other and may vary only as for the name of the controlled plant device. In such case, a PRESENTER-type object is very useful which is displayed close to this plant device.

In the presented example the role of the pointed object performs a PICTURES-type object, which displays the state of a valve. A transparent BUTTON-type object is superimposed over it to perform OPEN MASK -type actions. In order to display the given "control panel" diagram, click the mouse on the valve area. Doing so, the PRESENTER object will display an indicator pointing the valve. Closing the "control panel" diagram deletes the PRESENTER objects.

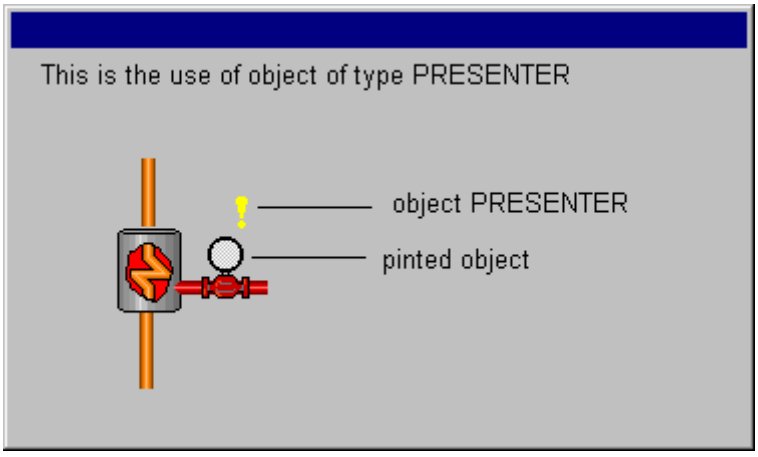


Figure 18-46. The PRESENTER Example.

Dimensions

The dimensions of the PRESENTER object are fixed (as determined on the basis of the bitmap size) and cannot be altered.

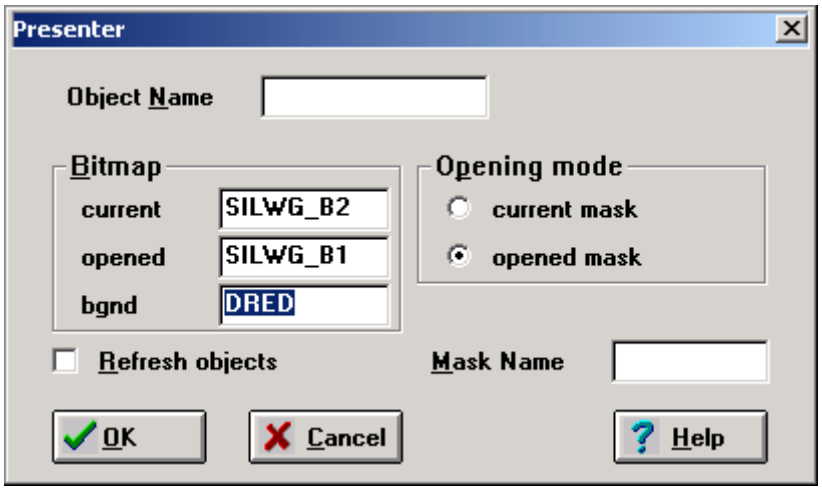


Figure 18-47. The PRESENTER Object Parameterization Window.

The PRESENTER object is displayed after a certain diagram is displayed. The mask (diagram) name is to be specified in the object parameter setting window. PRESENTER may be displayed if the diagram is either opened or opened and current at the same time.

NOTE *The opened diagram may be superimposed by another diagram thus becoming invisible. The current diagram is a lately selected one. The current diagram is always visible. If the current diagram performs the role of a "control panel", as long as it is a current diagram it can be used to send the controlled variables from it.*

In order to delete, *PRESENTER* object superimposes the old object with a bitmap colored with the same background color. If the *PRESENTER* object superimposes completely or partially an another object, deletion of it should cause reappearing of the superimposed object. This function depends on the contents of the "Refresh objects" box".

Parameters

Object Name - this text box is used to put into it the optional name of the object. If the name of the object will not be defined, its type together with coordinates will appear on the list of objects.

Bitmap – Current - a text box used to specify the bitmap (picture) name to be displayed on opening the diagram of the given name and selecting it so becomes a current diagram. The bitmap name may be either entered in the dialog box or selected by clicking the right mouse button. A

window appears for previewing and selecting the bitmaps belonging to the **asix** system bitmap pool.

Bitmap - Opened - a text box used to specify the name of a bitmap (picture) to be displayed on opening the diagram of the given name. The bitmap name may be either entered in the dialog box or selected by clicking the right mouse button. A window appears for previewing and selecting the bitmaps belonging to the **asix** system bitmap pool.

Bitmap - Bgnd - a text box used to specify the name of a bitmap (picture) to be displayed as a background, which will remove the previous bitmap. The bitmap name may be either entered in the dialog box or selected by clicking the right mouse button. A window appears for previewing and selecting the bitmaps belonging to the **asix** system bitmap pool.

Mask Name **-** a text field used to enter the mask name, which, when opened (and possibly selected) should cause starting the *PRESENTER* object up. The diagram name may contain „?" and „*" characters, being wildcards for, accordingly, single character or a string of characters.

Opening mode-Current mask - a radio button used to display the bitmap, the name of which is specified in the ***Bitmap-current*****,** if a diagram which name is given in the ***Diagram Name*** box has been opened and selected**.**

Opening mode-Opened mask - a radio button used to display the bitmap, the name of which is specified in the ***Bitmap-opened*****,** if a diagram which name is given in the ***Diagram Name*** box has been opened**.**

Refresh objects - a check box used to force reappearing of the object superimposed by the *PRESENTER* object. This is done after the ***Bitmap-bgnd*** is displayed.

18.4.26. RECTANGLE Object

Dimensions
Object Parameters
- Object Name
- Line Color
- Fill Color
- Shadow
- Line thickness
- Depth
- Upper Corners
- Lower Corners
- Fill Inside
- Window Background
- Draw Square
- Draw Frame
- Frame Color

The RECTANGLE object enables a rectangle to be displayed in the diagram. This is a static object. It is possible to declare its size and color. A square may de drawn as a special case of the rectangle.

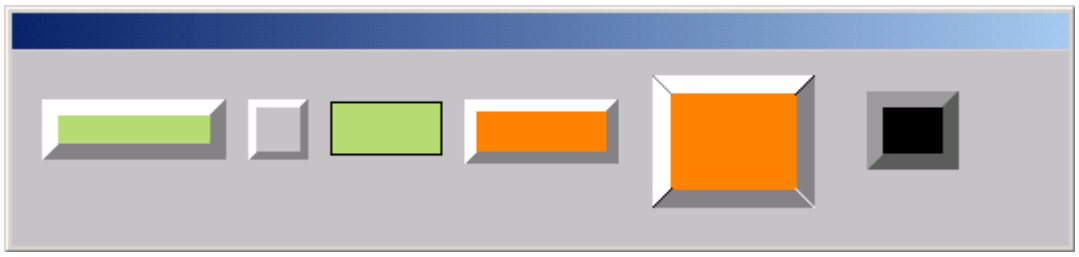


Figure 18-48. The RECTANGLE Examples.

Dimensions

The height and width are specified with use of the mouse.

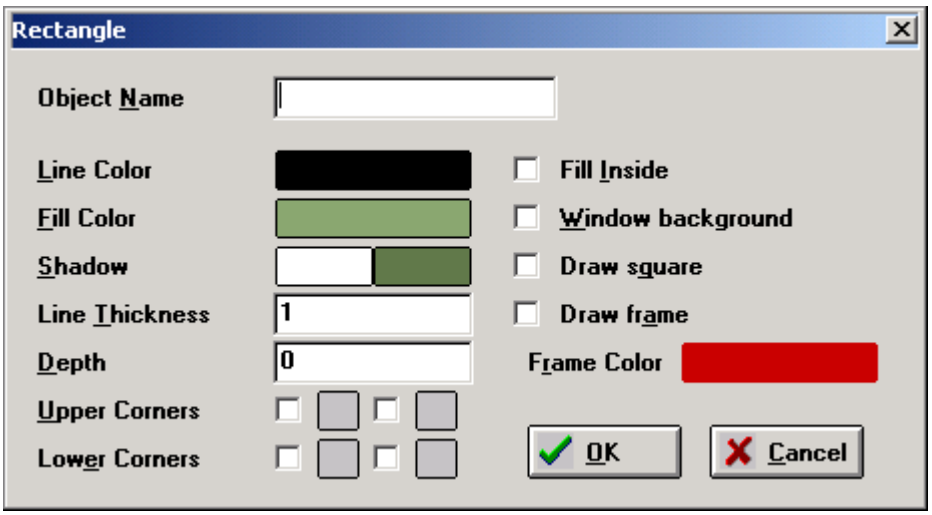


Figure 18-49. The RECTANGLE Object Parameterization Window.

The rectangle can be drawn with the shading effect and this is decided by the *Depth* parameter. It is possible to define the color of highlight individually (left and upper side of the rectangle) and also the color of the shadow (left and lower side). If light colors are chosen for the highlighting, and dark colors for the shadow the convexity effect will be obtained. If dark colors are chosen for the highlighting, and light colors for the shadow, then concavity effect will be obtained. Additionally, a frame with rounded corners can be drawn in, and also the corners in suitable color. It enables drawing "3D" rectangles being the background for other objects. The rectangle without shading can be received, if depth parameter value is equal to zero.

Parameters

Object Name - this text box is used to enter into it the optional name of the object. If this name is not defined, its type together with coordinates will appear on the list of objects.

Line Color - a color box used to specify the color of the line surrounding the rectangle after the color selection window is displayed.

Fill Color - a color box used to specify the color of the rectangle inside after the color selection window is displayed.

Shadow - color boxes used to specify colors of the shadow surrounding the rectangle (if the depth parameter is non-zero). Colors of the upper left shadow and of the lower right shadow are set respectively. By a proper choice of colors, effects of concavity/convexity are obtainable. Colors are selectable after the color selection window is displayed.

Line thickness - a numerical box used to specify the thickness of the line surrounding the rectangle (Default = 1).

Depth - a numerical box used to specify whether a rectangle is to be drawn with the depth effect (if the value is non-zero) or not. (Default = 0). The value of this box determines the shadow width.

Upper Corners - a set consisting of two check boxes and two color boxes to specify the parameters of the upper left and right corners respectively. The check box determines whether the corner is to be drawn or not whilst the color box defines the line color.

Lower Corners - a set consisting of two check boxes and two color boxes to specify the parameters of the lower left and right corners respectively. The check box determines whether the corner is to be drawn or not whilst the color box defines the line color.

Fill Inside - a check box used to fill the rectangle inside.

Window Background - a check box used to force drawing of the inside with the color of the diagram background.

Draw Square - a check box used to force drawing of a square.

Draw Frame - a check box used to force drawing of a frame (for rectangles with shadowing).

Frame color - a color box used to declare the frame color after the color selection window is displayed.

18.4.27. RECTANGLES Object

Dimensions
Object Parameters
- Object Name
- Parameters
- Sending
- State
- Line
- Bgnd
- Blink
- Coding
- Variable
- Bit mask

Coding Example

The RECTANGLES object is a modification of the RECTANGLE object to enable changing the colors of the drawn rectangle in function of the monitored variable state. This is a dynamic object (if it is a multistate object). Optionally, RECTANGLES object may also be selectable and may serve to send a determined value to the controlled variable. Sending of value may be protected by a password.

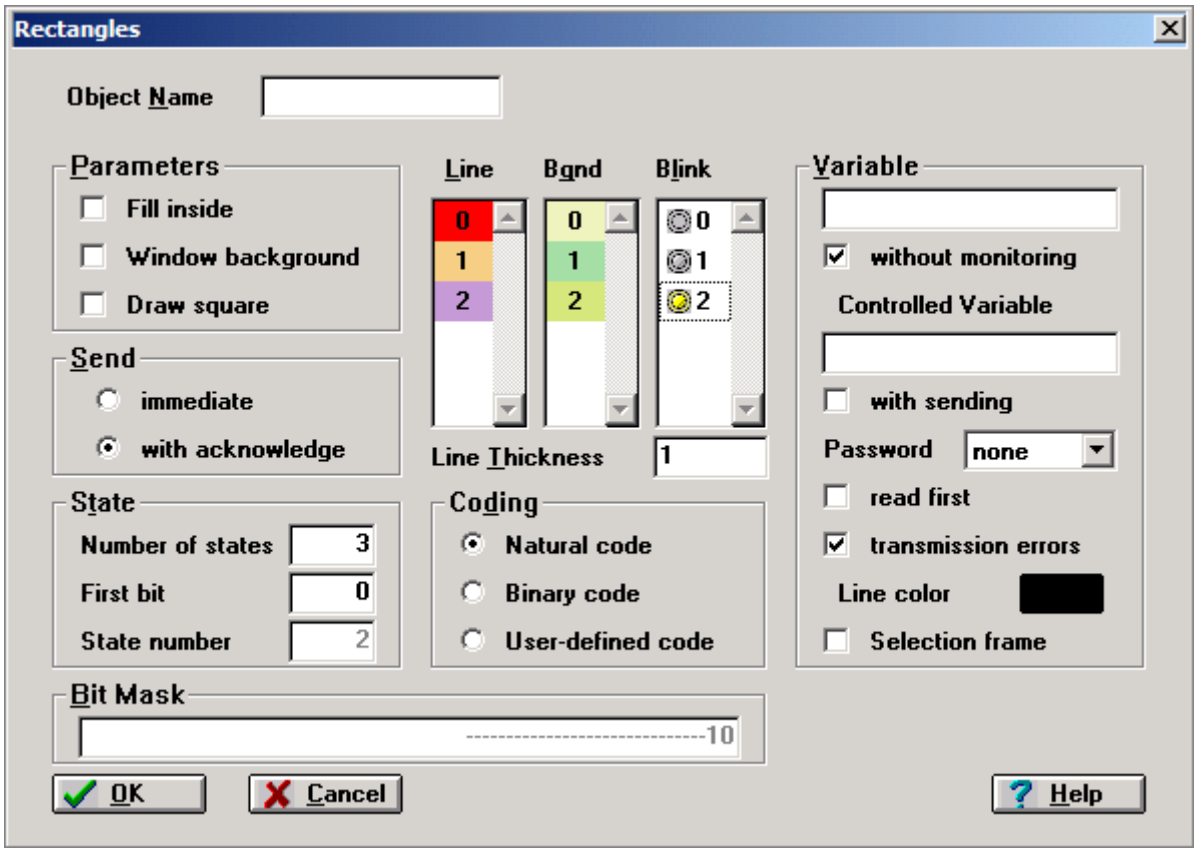


Figure 18-50. The RECTANGLES Object Parameterization Window.

Dimensions

The dimensions of the rectangle are set with use of mouse.

By default, the object has one state only thus it is a static object. Changing the color signals transmission errors.

Parameters

Object Name - this text box is used to enter into it the optional name of the object. If the name of the object is not defined, its type together with coordinates will appear on the list of objects.

Parameters-Fill inside - a check box used to fill the rectangle inside with the declared color.

Parameters-Window Background **-** a check box used to force drawing the rectangle inside with the background fill color of the window background irrespective of the declared inside color.

Parameters-Draw Square **-** a check box used to force drawing a square.

Send Immediate - a radio button used to declare immediate transfer the value, after the corresponding rectangle is selected, by clicking the mouse button on the object box while being under the diagram refresh mode (only possible if the object has been set as controllable one). This type of sending is reasonable if the object has two states otherwise the object state previewing as resulted from the selection of the successive states will cause sending of a series of values. The value by object is the number that corresponds to the state number of object.

Send with Acknowledge - a radio button used to declare sending the value after the corresponding rectangle is selected by clicking (many times) the mouse button on the object box while being under the diagram refresh mode (only possible if the object has been set as controllable one) and then by acknowledging. Acknowledge of sending can be made by either clicking the button "*Grey* +" or executing appropriate actions. This type of sending is reasonable if the object is a multistate object. The value by object is the number that corresponds to the state number of object.

State - Number of States - a numeric box that enables declaring a number of states of the object. By default the object has 1 state and then it is the static object. Maximum number of states is 33 for natural coding, 64 for binary coding and user-defined coding. The individual states correspond with values that are result of the chosen coding mode.

State - First Bit - a numerical box is provided to declare the first (least significant) bit of the monitored variable, the value of such variable is displayed as one of many declared rectangles. The contents of this box (valid for both natural and binary coding only) permit for an unambiguous determination of the position of the group of the bits which control the rectangles as displayed in the word. The first bit corresponds to the LSB whilst the group size corresponds to the coding method.

State -State Number - a numerical box is provided to display only the number of the state selected. Selection is made by clicking on the appropriate place of one of two color lists.

Line - color list corresponding to the surroundings of individual rectangles. To modify the color, choose a suitable item from the list and press *Enter*. The color selection window will appear. Trying to declare the name of a non-existing rectangle will fail. The state numbers (from 0 up to n) are displayed black in the color list.

Bgnd - a color list corresponding to the backgrounds of individual rectangles. To modify the color, choose a suitable position from the list and press *Enter*. The color selection window will appear. Trying to declare the name of a non-existing rectangle will fail. The state numbers (from 0 up to n) are displayed black in the color list.

Blink - opening the blinking attribute configuring window.

Coding - Natural Code - a radio button provided to choose a natural code (for coding methods – see chapter after Parameter List).

Coding - Binary Code - a radio button provided to choose a binary code (for coding methods - see chapter after Parameter List).

Coding - User-Defined Code - a radio button provided to choose the user-defined code (for coding methods - see chapter after Parameter List).

Variable - a text box provided to specify the name of the monitored variable. Specifying the name is not necessary if the option ***without monitoring*** has been selected. Clicking the right mouse button on this box opens a window containing the list of the available variables to make selection.

Controlled Variable - the text box, which allows specifying the name of control variable. Specifying this name is necessary, if ***with sending*** option has been selected. By pressing the right mouse button in this field you can open the window with the list of available variables and make the choice in it. It is possible to specify the name of controlled data, using notation with **#** character. If in the field of variable name the

#_ characters with suffix are entered, then during the application's run it will be treated as the variable with name containing the name of monitored variable instead of the # character.

Without Monitoring **-** a check box provided to set the object parameters in such manner that the displayed state is not dependent on any system variable and serves only to indicate the value to be sent.

With Sending **-** a check box provided to set the object parameters in such manner that the value corresponding to its state could be sent to the system variable. Sending of the value requires the object to be selected, e.g. by clicking the mouse; what will be done afterwards depends on the sending mode.

Password - password combo box that protects sending the controls (four levels of password can be selected, no password is the default setting).

Read First **-** a check box provided to force reading before sending of the control variable. In such case, there is written to the control variable a logic sum of the sent value and of the given variable value as read from the controller with the bit group, which has been reset by the object.

Transmission error **-** a check box provided to warn about transmission errors (the object is crossed). Default option.

Line Color - a color box of the crossing line which signals transmission error occurrence. Selection of the color box causes displaying of the color selection window.

Bit Mask - a numerical box provided to display the monitored variable (the variable, at the same time, which can be sent from the object if the parameters of this latter have been set as ***with sending*****),** which corresponds to the given state. Under user-coding, the diagram can be set by inserting a hexadecimal or binary code corresponding to the given state, possibly indicating the user-defined bits with „-".

The ***'Coder'*** window is used to define the states in user-defined coding mode:

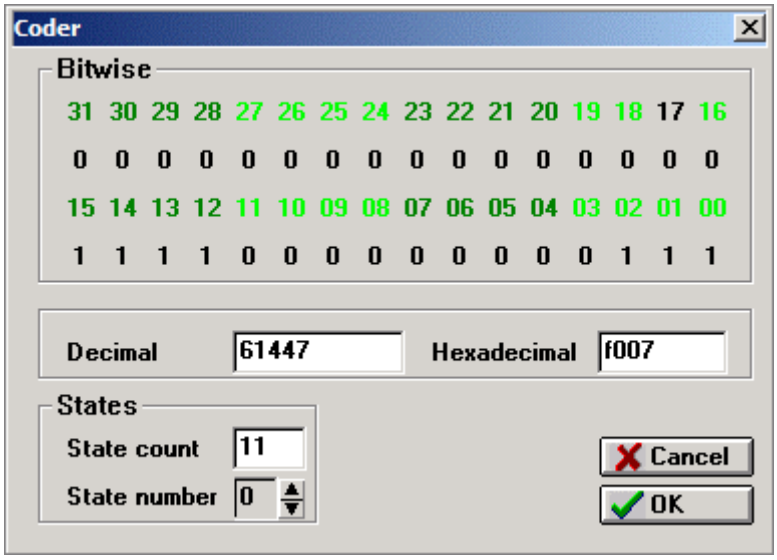


Figure 18-51. The 'Coder' Window.

The number of states recognized by the object is passed in *State count* field. This value may be also set in the main window for object defining. The state the code of which should be defined is selected in *State number* field. While setting the number in *Bitwise* frame, the current state coding is displayed - to change it, one should click with mouse on a bit value. Successive click operations cause change of the value in the following sequence:

- – bit value is unimportant
1 – bit has to be set
0 – bit has to be set to zero

Sate coding is automatically stored in the moment of choosing the other state.

The *Decimal* and *Hexadecimal* fields display decimal and hexadecimal value of defined state.

Selection Frame **-** a check box permitting additional displaying the so called selection frame (object contour) after the object is selected during the diagram refresh. The frame helps to know quickly which object is the last selected one.

During the refresh operation, the object can be selected using either the mouse cursor or the *Tab* key (*Shift*+*Tab*). After the selection is made, choose the given object state (which is signaled by a change of the picture) with the mouse cursor or the *Enter* key. Sending of the state is done automatically after selection/acknowledge is made. To acknowledge, press the button *Grey+* or carry out appropriate actions. If you don't want to do selection, either press *Esc* key or click the right mouse button.

There is a possibility to use the conversion function, the values of which are the indexes of successive rectangles as well as to use the another conversion function, which changes the object state number to the sent value when the option *With control* is active. The decoding of a fixed-point monitored variable

value (and coding of possibly sent values as well) using one of three object-embedded strategies is possible too.

Natural Coding

The group of bits is checked. Number of bits is equal to the *number of states*, but number of the LSB defines "offset". The group of bits can consist only one value of "1". Number of the state is calculated as position of the "1" within the group of bits (the LSB position is number one). The zero state corresponds with the group of all zeros.

Binary Coding

The group of bits is checked. Number of bits is equal to the binary logarithm of *number of states*, but number of the LSB defines "offset". Number of state is calculated as content of the tested bit group.

User-Defined Coding

The total word is checked. The "offset" parameter is ignored. The user assigns a string of ones and zeros (the *don't carry* states can be masked by "-") to each state. Number of state is calculated as the first of states, that the user assigned the same bits on the appropriate positions (regardless to the masked bits).

For the objects which are set *with sending*, the option *read first* may be used. In such case, there is written to the controlled variable a logic sum of the sent value and of the given variable value as read from the controller with the bit group, which has been reset by the object.

EXAMPLE

The object has four states and the binary coding has been chosen. To code four binary states (00,01,10,11) a two-bit group in the word is necessary. The first bit of the group determines the contents of the field ***State-First bit***. If it equals for example to 4, then the state of only the fifth and sixth bits will affect the object state. The remaining bits have no influence on displaying of the object what is illustrated below.

```
----------xx----
```

("-" denotes a non-affecting bit, "x" denotes a bit which affects the object state).

If you define four rectangles e.g. yellow, green, blue and red (consecutively) which corresponds to these four states, the yellow rectangle corresponds to the state "00", the green rectangle corresponds to the state "01" and so on. With writing, if you choose the red rectangle, the corresponding state "11" of the fifth and sixth bits i.e. the decimal number 48 is sent to the variable. When the ***read first*** box is activated, this mechanism is somewhat different - at first the given variable is read and then the fifth and sixth bits are reset, the state corresponding to the color of the rectangle selected being overwritten to these bits.

```
1000000000100000     / the read variable
1000000000--0000     / the read variable with reset bits „-"
```

```
----------11----          / bits which are set by the object
1000000000110000          / the sent variable
```

18.4.28. REFRESH TEST Object

Dimensions
Object Parameters
Object Name

The REFRESH TEST object is designed to perform tests. It allows knowing what is the refreshing frequency value for the given diagram (in other words, how frequently the objects may be re-written on the diagram). The test result is displayed. The test result may give indications of a slow-operating computer or of too many objects placed on the diagram.

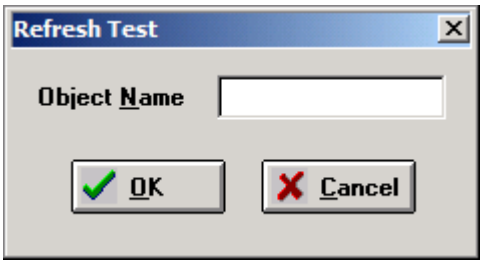


Figure 18-52. The REFRESH TEST Object Parameterization Window.

Dimensions

The dimensions of the REFRESH TEST object result from the number of characters and from the default font (dialog).

The object serves to verify diagrams during their testing. Two numbers having the **nn (nn)** format are displayed on the diagram. These numbers are to be interpreted as a number of diagram refreshing operations in 15-second period. The first number (which is always incremented by itself) is the current test result. The number in brackets is the previous test result. If the result is worse than 15 (i.e. less than 1 refresh per second) this is a bad result. The REFRESH TEST object should be eliminated in the applications.

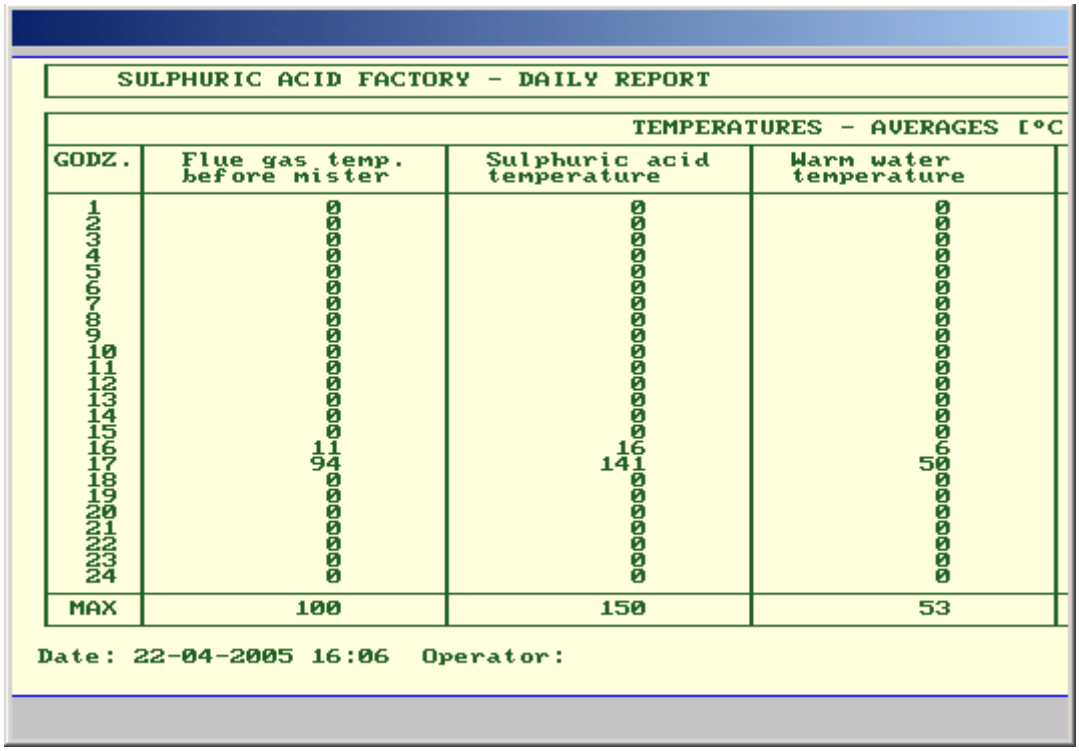
Parameters

Object Name - this text box is designed to enter the optional name of the object. If the name of the object will not be defined, its type together with coordinates will appear on the list of objects.

18.4.29. REPORT Object

Dimensions
Object Parameters
Object Name
Report Definition in the ASTER Language
Report Name
Calculating
Terms
Cycle
Outlook
Read
Write

The REPORT object enables a report (definition of which has been written in ASTER language) to be verified and calculated and a resulting report to be displayed over the process diagram. The reports are numerical arrays, which are as usual displayed as tables. Reports may be calculated cyclically. This object can also make a calculation on operator's request (after the object is selected). It is also possible to make automatic report calculations at determined time of the day (up to 8 time points a day).



SULPHURIC ACID FACTORY - DAILY REPORT

TEMPERATURES - AVERAGES [°C

GODZ.	Flue gas temp. before mister	Sulphuric acid temperature	Warm water temperature
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	0	0	0
11	0	0	0
12	0	0	0
13	0	0	0
14	0	0	0
15	0	0	0
16	11	16	6
17	94	141	50
18	0	0	0
19	0	0	0
20	0	0	0
21	0	0	0
22	0	0	0
23	0	0	0
24	0	0	0
MAX	100	150	53

Date: 22-04-2005 16:06 Operator:

Figure 18-53. The REPORT Example.

Dimensions

Both the height and width are automatically set according to the report definition (on the basis of the contents of the format instruction and of the font selected). Numeric values written to the reports during the work with the

designer are simulated ones. In the case of calculation request, the operator may force the calculation by selecting the object. To make a cyclical evaluation it is necessary to determine the cycle duration. Since the report calculation is carried out during the refreshing cycle, reports should be calculated not longer than a dozen of seconds (to be verified individually). The calculation time is directly proportional to:

- the number of expressions to be evaluated (product of lines and columns of the report) and
- the number of measurements archived in the reported period (product of the report period and the sampling frequency of variables).

The reports planned as requiring longer calculation times should be planned as disk reports.

Definition of the ASTER language should be written in the box below the object name. The best solution, however, is to import a ready-to-use definition from the disk (after selecting the report name) using the READ function with a possible modification to be made in the editing box. The report definition is saved within the object. The report definitions can be verified by clicking on the diagnostic box situated below the definition box. If the report is erroneous then an error message is displayed in the diagnostic box with the line containing the error. If a syntax error is made in the error line, a question mark (?) appears. If the READ function is used for an empty report name, then an empty report is written in the editing box.

A modified report may be written by means of the WRITE function. Writing is made to the report the name of which is indicated in the report name box (report's name is not a filename). If the WRITE function is used for an empty report name, then the filename is automatically generated.

It is possible to choose the font of the object report. Contrary to the remaining texts, there are available three fonts only: **ROM_8x8, ROM_8x14** and **ROM_8x16.** These fonts feature a fixed character width and have the frame components defined as characters. Moreover, the fonts ensure compatibility to the earlier **asix** versions (running under DOS).

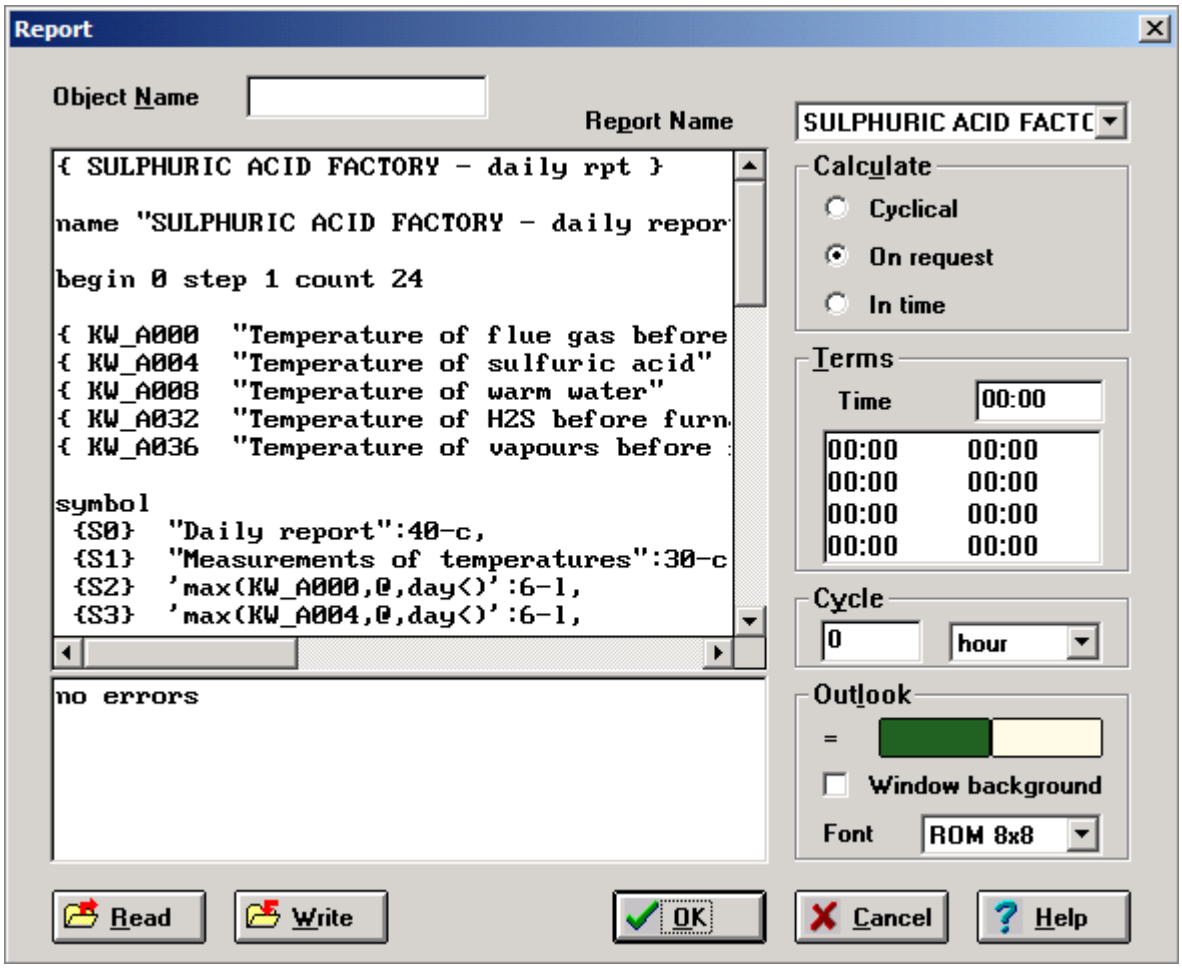


Figure 18-54. The REPORT Object Parameterization Window.

Parameters

Object Name - this text box is used to enter into it the optional name of the object. If the name of the object will not be defined, its type together with coordinates will appear on the list of objects.

Non described boxes **-** a text box provided to insert an ASTER language report (main box of the object name) and to display the diagnostic messages (the box below). Clicking the right button in this box causes opening of the ***Expression Generator*** window to permit for expression generation to be made by an user which does not know the ASTER language syntax.

Report Name **-** a selection box provided to select a report to be read/written. These reports (strictly their definitions in the *.r files) should be placed in the directory specified in the section of the [REPORT] of the *.ini file. This box may be

	empty - to indicate that there is no report definition.
Calculate-Cyclical	- a radio button provided to force cyclical evaluation of an expression by the object.
Calculate-On request	- a radio button provided to force evaluation on operator's request (by clicking on the object box).
Calculate-In time	- a radio button provided to force evaluation according to the list of the terms.
Terms - Time	**-** a text box provided to specify a term (time) of report calculation. Below of this box there is a list of terms (up to 8). Each term can be inserted to the list by pressing of Enter key.
Cycle	**-** a numerical box and a combo box to determine the refreshing cycle of the object by specifying the number of time slices and the unit. This box is important for the cyclical mode of operation.
Outlook -=	**-** a color boxes provided to specify the character color (left box) and the background color (right box) of the displayed report.
Outlook - Window background	**-** a check box provided to force the background color being the same as the window color.
Outlook - Font	**-** a combo box provided to choose one of three available characters for report writing.
Read	**-** a button provided to read the report as set in the ***Report Name*** box to the editing box.
Write	**-** a button provided to write in the report situated in the edit box to the file, which comprises the report, the name of which is set in the ***Report Name*** box.

18.4.30. SELECTOR Object

Dimensions
Example
Object Parameters
- Object Name
- Button
- Sending
- Text List
- State
- Bit Mas
- Coding
- Variable
- Parameters
- Justify

The SELECTOR object is a combination of a button with a window displaying the explanatory text. Text identifiers may be used instead of explanatory texts. The identifiers are declared in ***Text parameters*** parameter in application

configuration file with use of Architect: Architect > *Fields and Computers* > *Masks* module > *Text parameters* tab.

This approach permits the use of the same mask for different applications (which in their descriptive parts).

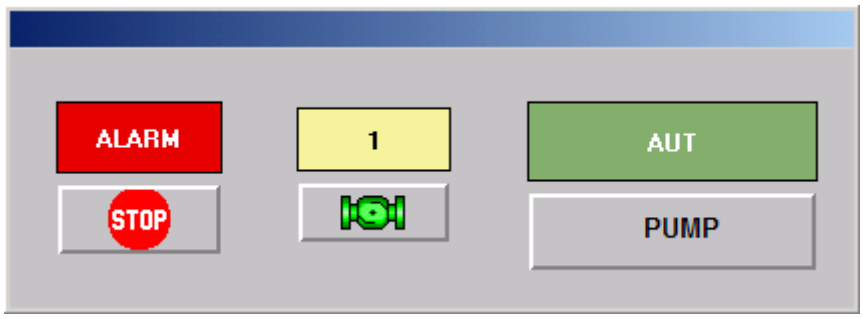


Figure 18-55. The SELECTOR Examples.

This is a selectable object. Selection of a button causes the description to be displayed, corresponding to the controlled variable to be sent. Sending may be accomplished either on button clicking or after acknowledgement is made. Sending the controlled variable can be password-protected. The object is also a dynamic object, which displays a state corresponding to a determined variable monitored by the controller in the text window. Certain (values declared as "selection protected") can be sent from the SELECTOR object by clicking either the mouse button or *Enter* key and additionally *Ctrl* key. Thus, a protection is ensured to not send unintentionally certain values. The other states (for example unused ones) may feature "selection disabling" to disable their displaying during previewing of the state list. The SELECTOR object buttons may be transparent (but the text window remains). Then, another object is to be inserted in place of the selector button (e.g. BUTTON).

Dimensions

The dimensions of the SELECTOR object are determined on the basis of the text which describes the selector state, of the text which describes the button, of the font and of the size of possible bitmaps which create the button. In order to determine the size, previewing is made of all text displayable in the text window. Such determined object size is a minimum size, which may be enlarged, along with the button beneath using the mouse. Trying to reduce the object below its minimum size will fail.

The button description text is to be inserted into the list position pointing with the mouse cursor. Pointing of the text on the list causes displaying of its attributes in appropriate dialog boxes. The display attributes (font, centering, colors, access protection & disabling) can be modified. They will be assigned to the given text indicated in the list.

By default, the object has one state only. Crossing it with lines of specified color and thickness signals communication errors.

The object may be selected during refreshing operation using either the mouse cursor or the *Tab* key (*Shift*+*Tab*). After the selection is made, you can display the required state with either the mouse cursor or the *Enter* key. Protected

states can be displayed by pressing additionally *Ctrl* key. Sending the state is done automatically after selection/acknowledge is made. To acknowledge, press either the button *Grey+* or carry out appropriate actions. If you don't want to do selection, click either *Esc* or the right mouse button.

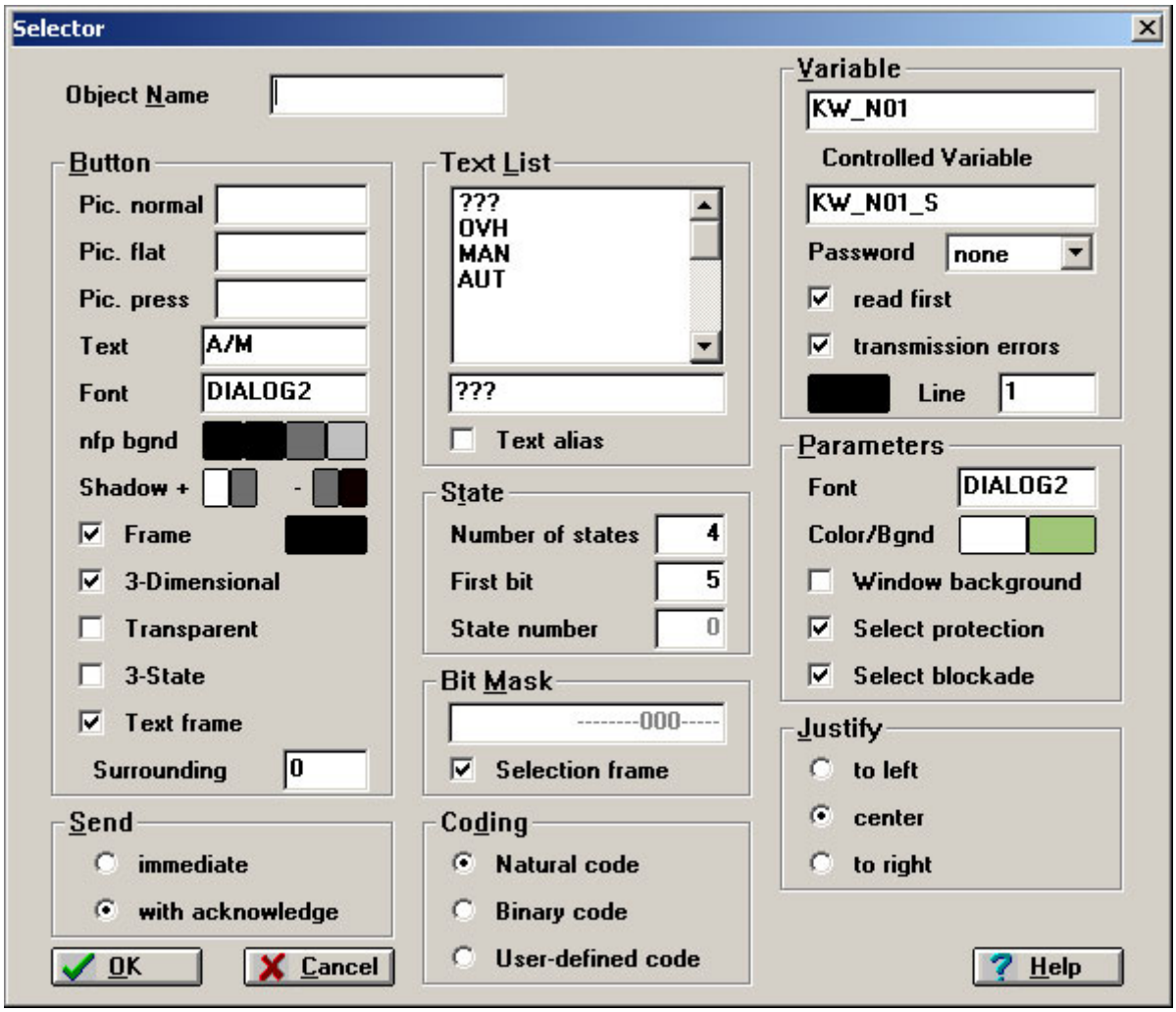


Figure 18-56. The SELECTOR Object Parameters Window.

Parameters

Object Name - this text box is used to enter into it the optional name of the object. If the name of the object will not be defined, its type together with coordinates will appear on the list of objects.

Button-Pic. normal - a text box, used for entering the name of bitmap, displayed on the unpressed button. The bitmap's name can be either entered in the dialog field or can be chosen by pressing the right mouse button. The window appears then, enabling browsing and selection of bitmaps being part of **asix** system maps pool.

Button-Pic. flat - a text box, used for entering the name of bitmap, displayed on the flat button. The bitmap's name can be either entered in the dialog field or can be chosen by pressing the right mouse button. The window appears then, enabling browsing and selection of bitmaps being part of **asix** system maps pool. The flat button status may occur only in case of selecting the **tri-status** window, described below.

Button-Pic. press - a text box for specifying the name of bitmap displayed on the pressed button. The bitmap's name can be either entered in the dialog field or can be chosen by pressing the right mouse button. The window appears then, enabling browsing and selection of bitmaps being part of **asix** system maps pool.

Button - Text **-** a text box to specify the caption of the selector button. It is an alternative to the use of the bitmaps declared in the ***Button - Picture*** box.

Button-Font **-** a text box provided to specify the font of the button caption characters. The default font is Dialog. Clicking the right mouse button on this box causes activation of the font selection window.

Button-Bgnd **-** color boxes provided to specify the button caption color (left box), the text color for the button-as-pressed (middle box) and the background (right box). After selection is made, the window of color selection is displayed.

Button-Shadow + **-** color boxes provided to specify the button caption color (left box), the text color for the button-as-pressed (middle box) and the background (right box). After selection is made, the window of color selection is displayed.

Button-Shadow **-** color boxes provided to specify the illumination feature (left box) and the color of the shadow of the button-released (right box). Illumination is related to the button upper and left sides whilst the shadow is related to the lower and right sides. After selection is made, the window of color selection is displayed.

Button - Frame **-** a check box and a color box to specify whether the button should be provided with a frame and to specify the frame color.

Button - 3-Dimensional **-** a check box, which permits drawing of so called three-dimensional buttons.

Button - Transparent **-** a check box to draw transparent buttons. In this case, the proper buttons should be created by means of other objects.

Button – 3-State - a check box that allows third flat state of the object.

Button -Text Frame **-** a check box and color box provided to specify, whether the text frame (situated above the button) should be drawn.

Button - Surrounding **-** a numerical box provided to specify the surrounding of the button in terms of pixels (Default = 0). The surrounding permits to enlarge the selection sensitive zone (essential for small buttons).

State - Number of States - a numerical box provided to declare the number of states of the object. By default, the object has 1 state i.e. it is a static object. Maximum number of states is 17. The values associated with the state result from the coding method selected.

State - First Bit - a numerical box provided to declare the first (least significant) bit of the monitored variable, the value of such variable is displayed as one of many selector states. The contents of this box (valid for both natural and binary coding only) permit for an unambiguous determination of the position of the group of the bits, which control the states as displayed in the word. The first bit corresponds to the LSB whilst the group size corresponds to the coding method.

State - State Number - a numerical box only to display the number of the state selected. Selection is made by clicking on the appropriate item of the ***Text List***.

Text List - list of text corresponding to the individual states of the object. To change text, select it on the list, insert a new text in the box below and press *Enter*. Trying to declare a text for a non-existing state will fail.

Text alias - a check box provided to inform that in the text list there are only the names of text equivalents, whilst the real texts are situated in the application *.xml file in ***Text parameters*** parameter (declared with use of Architect program > *Fields and Computers* > *Masks* module > *Text parameters* tab).

Bit Mask - a numerical box provided to display the monitored variable (provided when the ***Without monitoring*** option has not been used) and the variable which can be sent from the object corresponding to this state.

Selection Frame - a check box permitting additional displaying of the so called „Selection Frame" (Object contour) after the object is selected during the diagram refreshing. The frame helps to know quickly which object is the last selected one.

Coding - Natural Code - a radio button provided to choose a natural code (for coding methods - see chapter after Parameter List).

Coding - Binary Code - a radio button provided to choose a binary code (for coding methods - see chapter after Parameter List).

Coding - User-Defined Code - a radio button provided to choose a user-defined code (for coding methods – see chapter after Parameter List).

Send Immediate - a radio button provided to declare immediate sending of the value, after the corresponding state is selected, by clicking the mouse button on the object box while being under the diagram refresh mode. This type of sending is reasonable if the object has two states otherwise the object state previewing as resulted from the selection of the successive states will cause sending of a series of values. The value by object is the number that corresponds to the state number of object.

Send with Acknowledge - a radio button provided to declare sending of the value after the corresponding descriptive text is selected by clicking (many times) the mouse button on the object box while being under the diagram refresh mode and then by acknowledging. Acknowledge of sending can be made by either clicking the button "*Grey +*" or executing appropriate actions. This type of sending is reasonable if the object is a multistate object. The value by object is the number that corresponds to the state number of object.

Variable - a text box provided to specify the name of the monitored variable. Specifying the name is obligatory. Clicking the right mouse button on this box opens a window containing the list of the available variables to make selection.

Controlled Variable -a text box that allows specifying the name of controlled data. Specifying the name is obligatory. By pressing the right mouse button in this field a window with list of available variables can be displayed, and you can make a choice in it. It is possible to specify the name of status data, using notation with **#** character. If in the field of variable name the #_ characters with suffix are entered, then during the application's run it will be treated as the variable with name containing the name of monitored variable instead of the # character

Password - a password combo box that protects sending the controls (four levels of password can be selected, no password is the default setting).

Read First **-** a check box provided to force reading before sending the controlled variable. In such case, there is written to the controlled variable a logic sum of the sent value and of the given variable value as read from the controller with the reset bit group which has been set by the object.

Transmission error **-** a check box to warn about transmission errors (the object is crossed). Default option.

Line - a color box of the crossing line signalling transmission error occurrence. A numerical box

provided to decide on line width. Selection of the color box causes displaying of the color choice window.

Parameters-Font **-** a text box provided to specify the font of the characters used to display the text describing the selector state. It is possible to set individually this box for each text. The default font is Dialog. Clicking the right mouse button on this box causes activation of the font selection window.

Parameters-Color/Bgnd **-** color boxes which serve to specify the selector descriptive text color (left box) and the text background color (right box). After selection of this box is made, the window of color selection is displayed. It is possible to set individually this box for each text.

Parameters-Window Background **-** after selection of this check box is made, the window background color is selected as a text background color. It is possible to set individually this box for each text.

Parameters-Select Protection **-** a check box provided to set a so called "select protection" of each state, what is reduced to the necessity of clicking with the *Ctrl* key additionally (apart from clicking the mouse button or the *Enter* key) to accomplish an action. It is possible to set individually this box for each text.

Parameters-Select blockade **-** a check box provided to set a so called "select disabling" of each state, what is reduced to disable certain states to block their sending. It is possible to set individually this box for each text.

Justify-to left - a radio button provided to force left alignment of the object descriptive text. It is possible to set individually this box for each text.

Justify-center - a radio button provided to force centering of the object descriptive text. It is possible to set individually this box for each text.

Justify-to right - a radio button provided to force right alignment of the object descriptive text. It is possible to set individually this box for each text.

There is a possibility to use a conversion function, which values are the indexes of successive texts as well as to use an another conversion function which changes the object state number with the sent value. Possible also is decoding of a fixed-point monitored variable value (and natural coding of possibly sent values as well) using one of three object-embedded strategies.

Natural coding

The group of bits is checked. Number of bits is equal to the *number of states*, but number of the LSB defines "offset". The group of bits can consist only one value of "1". Number of the state is calculated as position of the "1" within the group of bits (the LSB position is number one). The zero state corresponds with the group of all zeros.

Binary Coding

The group of bits is checked. Number of bits is equal to the binary logarithm of *number of states*, but number of the LSB defines "offset". Number of state is calculated as content of the tested bit group.

User-Defined Coding

The total word is checked. The "offset" parameter is ignored. The user assigns a string of ones and zeros (the *don't carry* states can be masked by "-") to each state. Number of state is calculated as the first of states, that the user assigned the same bits on the appropriate positions (regardless to the masked bits).

For the objects which are set *With control*, the option *Read first* may be used. In such case, there is written to the controlled variable a logic sum of the sent value and of the given variable value as read from the controller with the bit group, which has been reset by the object.

EXAMPLE

The object has four states and the binary coding has been chosen. To code four binary states (00,01,10,11) a two-bit group in the word is necessary. The first (smallest) bit of the group determines the contents of the field ***State-First bit***. If it equals for example to 4, then the state of only the fifth and sixth bits will affect the object state. The remaining bits have no influence on displaying of the object what is illustrated below.

```
----------xx----
```

("-" denotes a non-affecting bit, "x" denotes a bit which affects the object state)

If you define four texts e.g. "???", "MAN", "AUT" and "REM" (consecutively) which correspond to these four states, the "???" text corresponds to the state "00", the "MAN" text corresponds to the state "01" and so on. With writing, if you choose the "REM" text, the corresponding state "11" of the fifth and sixth bits i.e. the decimal number 48 is sent to the variable. When the ***Read first*** box is activated, this mechanism is somewhat different - at first the given variable is read and then the fifth and sixth bits are reset, the state corresponding to the number of the text selected being overwritten to these bits.

```
1000000000100000    / the read variable
1000000000-—0000    / the read variable with reset bits „-"
----------11----    / bits which are set by the object
1000000000110000    / the sent variable
```

For practical purposes, it is possible to assign additionally the ***Selection blockade*** attribute to the state "00" ("???") whilst the ***Select protection*** attribute to the state "11" ("OVR"). Thus, it is impossible to select the "???" state and will become harder to select the "OVR" state (because of necessity of use of an additional *Ctrl* key).

18.4.31. SLIDER Object

Dimensions
Object Parameters
Object Name
Limits
Handle
Colors
Frame
Sending
Direction
Variable
Configuring from the VarDef

The SLIDER object enables a bar-type indicator fitted with a slider and corresponding to the measured variable to be displayed over the diagram. Position of the slider handle is controlled from the controlled variable value but may be also changed by the operator. SLIDER is a dynamic object. It is also a selectable object. After selection of the SLIDER type object it is possible to shift the slider handle. The moment of releasing the handle is accompanied by writing its position to the controlled variable and sending such position value to the controller. Thus, the second system variable related to the SLIDER object performs a role of a monitored and controlled variable at the same time. Sending of the value may be password-protected.

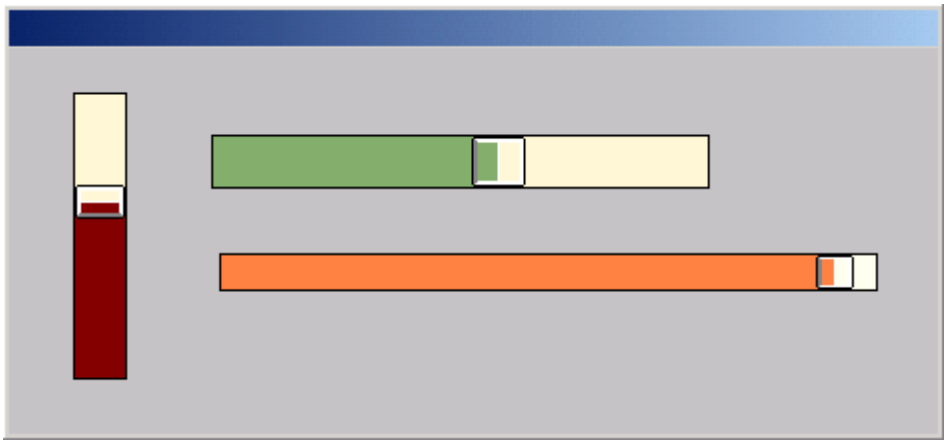


Figure 18-57. The SLIDER Examples.

Dimensions

The SLIDER object dimensions are determined with use of mouse.

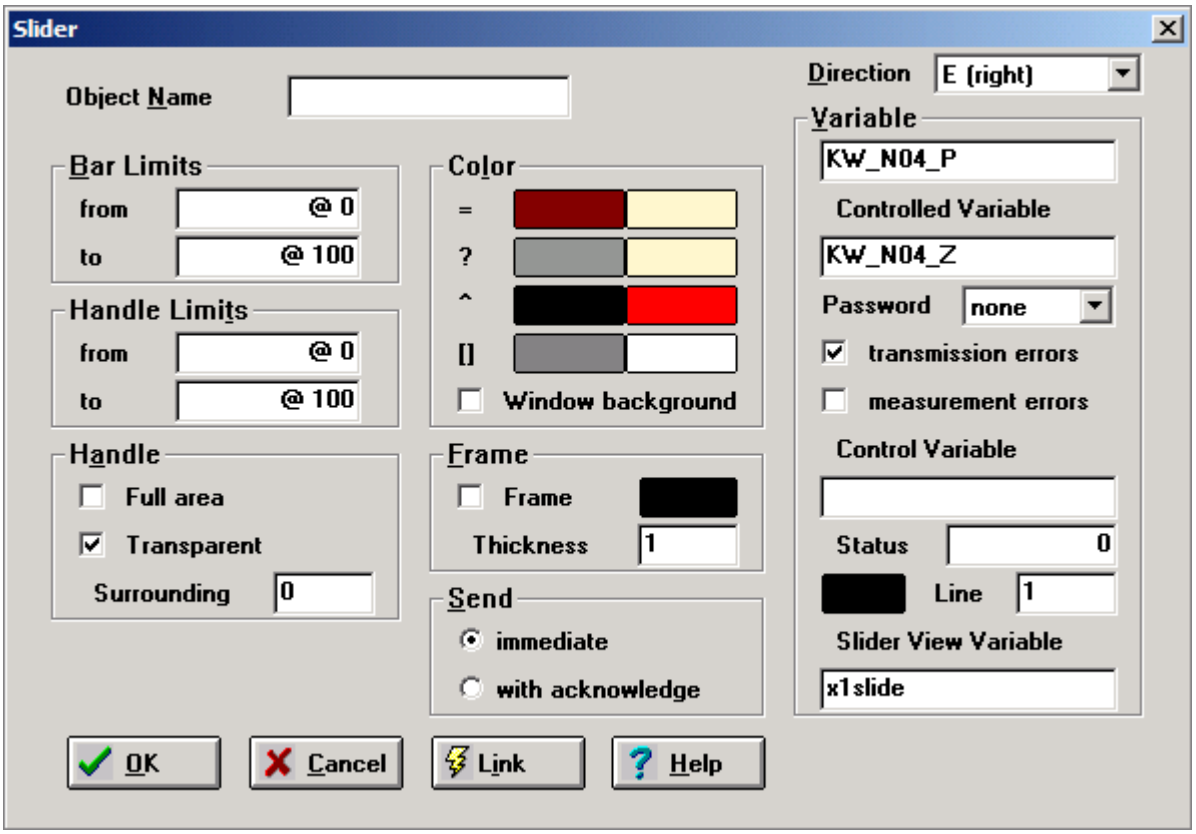


Figure 18-58. The SLIDER Object Parameterization Window.

Parameters

Object Name - this text box is used to enter into it the optional name of the object. If the name of the object will not be defined, its type together with coordinates will appear on the list of objects.

Direction - a combo box provided to determine the direction of the bar growth. The options are: N (up), E (right), S (down), and W (left). The default option is E.

Bar Limits -from - a numerical box provided to specify a numerical data corresponding to the lower limit of the displayed bar. The default option is 0.

Bar Limits -to - a numerical box provided to specify a numerical data corresponding to the upper limit of the displayed bar. The default option is 100.

Handle Limits -from - a numerical box provided to specify a numerical data corresponding to the lower limit of the displayed bar handle. The default option is 0.

Handle Limits -to - a numerical box provided to specify a numerical data corresponding to the upper limit of the displayed bar handle. The default option is 100.

Handle –Full area **-** a check box provided to specify whether the all object area should be sensitive on selection or handle zone only.

Handle -transparent **-** a check box provided to declare that handle should be „transparent" i.e. bar beneath the handle remains visible.

Handle -Surrounding **-** a numerical box to specify the surrounding of the handle in terms of pixels (Default = 0). The surrounding permits to enlarge the selection sensitive zone (essential for small buttons).

Color -= - color boxes provided to specify the basic color of the bar (left box) and the background (right box) for the displayed bar.

Color -? - color boxes provided to specify the basic color of the bar (left box) and the background (right box) for situation of error occurrence.

Color -^ - color boxes provided to specify the color of the bar handle (left box) and the color of the bar handle (right box) after its selection is made.

Color -[] - color boxes provided to specify the color of the bar handle illumination (left box) and the color of the bar handle shadow (right box).

Color - Window background - a check box provided to force the bar background color being the same as background window color.

Variable - a text box provided to specify a name of the measurement variable. Specifying the name is obligatory. By clicking the right mouse button on this box you can open a window containing the list of available variables to make selection.

Controlled Variable - a text box allowing specifying the name of controlled data. Specifying the name is obligatory. By pressing the right mouse button in this field, a window containing the list of available variables can be opened and you can make a choice in it. It is possible to specify the name of controlled variable using notation with **#** character. If in the field of variable name the #_ characters with suffix are entered, then during the application's run it will be treated as the variable with name containing the name of monitored variable instead of the # character.

Password - a password combo box that protects sending of the controls (four levels of password can be selected, no password is the default setting).

Transmission error **-** a check box provided to warn about transmission errors. When they occur, the bar is crossed along the line of specified parameters. Default option.

Measurement Errors **-** a check box provided to warn about measurement errors. When they occur, the bar is crossed with the lines of specified parameters. It requires specification of the ***Control Variable***.

Controlled Variable **-** a text box provided to specify the name of a control variable. If the ***Measurement errors***

option is selected, specification of this name is obligatory. Clicking the right mouse button on this box enables opening of the window of available variables and making selection.

Status - a text box provided to insert a hexadecimal mask. If the product of this mask and the control variable is different from zero then a measurement error occurs.

Line - a color box of the line, which crosses the object when errors occur and a numerical box, which is used to determine the line thickness. Selection of the color box causes displaying of the color selection window.

Slider View Variable - if the name of variable is entered, then the mechanism of automatic displaying and setting the value of view variable, during the operation of slider shift performed by an operator, will be switched on. The set value is equal to ones that would be send, when the operator released the slider. The value of view variable is less then the lower limit of the displayed bar, if the slider is not controlled by the operator. In *Slider View Variable* field the *#suffix* notation of variable name generation may be used. The name of view variable may be also set from the VarDef from the *PołożenieSuwaka* attribute.

Frame - Frame **-** a check box and a color box provided to specify whether the bar should be provided with a frame and to specify the color of the frame.

Frame - Thickness **-** a numerical box provided to specify the frame thickness (Default = 1).

end-immediate - a radio button enables to declare the immediate slider value sending (after slider release).

S-end-with acknowledge - a radio button enables to declare *sending with acknowledge* for the slider value.

REMARK The view variable should be declared in NONE channel. Its value may be displayed by NUMBER object. The view of variable (from *Slider View Variable* field) may be activate only at the moment of slider shift, if the proper attributes are set.

The slider handle can be made transparent. In such a case the handle borders along with the detail position dashes are displayed only. The portion of the bar beneath the slider remains visible. It is also possible to use a non-transparent handle made of position dashes, similar to a three-dimensional button. The object can be either sensitive on selection in "the entire range" (full area) or may require to select first the handle zone (with possibility of declaring the surrounding) before the handle is shifted. The slider may be surrounded with a frame.

The size of the handle is automatically set in function of the slider size. It is possible to change a color of the selected handle. When a transmission error occurs, the object is crossed transversely along the direction of bar movement with the lines having the declared parameters. When a measurement error occurs, the bar is crossed with the transverse lines having the parameters as

declared above. Additionally, when one error occurs (or two errors occur at the same time) the bar may change its colors.

Configuring from the Variable Definitions Database

The SLIDER object has been additionally provided with features enabling setting the following parameters from VarDef:

- range variation e.g. 0, 100 in accordance to the name of displayed variable,
- sent values range e.g. 0, 100 in accordance to the name of controlled variable,
- name and mask of control data.

In this purpose the ***Link*** button is used, which causes inserting the contents of a field in base, preceded with **@** character into all positions, that may potentially be set from the VarDef.

In the example below clicking on Link button inserts the contents of database fields for:

- limits of displayed variable – values 0 - 100
- limits of displayed variable's value– values 0 - 100
- control data – variable A002?
- control data mask – 4

18.4.32. SWITCH Object

Dimensions
Object Parameters
Object Name
States
Button
Zones
Variable
End of Repeating
Sending
Justify

The SWITCH object is a version of the button, which serve to send a value to controlled variable. Depending on version, it can have one or two buttons. The double-button version has separate buttons for switching ON and OFF respectively.

Sending may be password-protected.

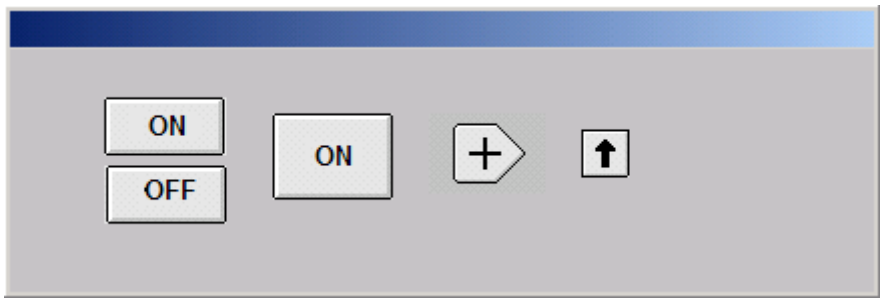


Figure 18-59. The SWITCH Examples.

In single-button version the key meaning is redefined depending on the status of monitored variable. If status of this variable corresponds to the status entered in ***Z pressed*** position, pressing the button will send the value or perform the action defined for this key at ***Z unpressed*** position. If the condition is not met, the object checks, if the status of monitored variable conforms to the second value. If it does, the value is sent or action from the condition, defined for the key at ***W unpressed*** position is performed. If none of the conditions is met, the operation defined for ***Z unpressed*** button is performed. Operation definition window for given key is opened after clicking on it:

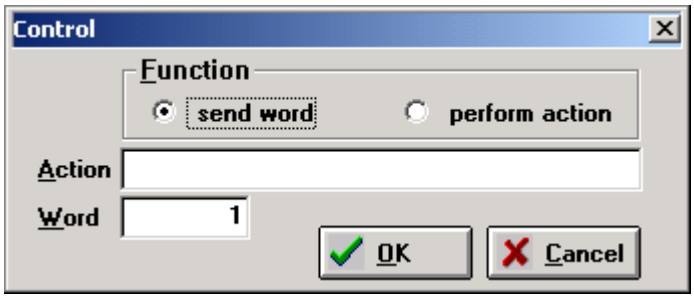


Figure 18-60. The 'Control' Window for Action Definition for the Button.

There may be an operator action or sending value, entered in the *Word* box.

The SWITCH object is selectable (it means that it sends controls) and dynamic, because its status changes, depending on the status of monitored variable, and on the key status (selected, pressed or not). If the switch is defined as tri-state, the button is flat until not selected. By selecting the object , by indicating it with the cursor or with the *TAB* key, the button becomes convex, pressed – after clicking the mouse at its area. After turning the tri-state option off the object may have only two statuses – pressed and unpressed.

In the SWITCH object, its buttons might be either a text type or bitmap type (it is not admissible to use both types of buttons at the same time). The button can be transparent. In such a case, another object (for example BUTTON or PICTURES) may be superimposed on it.

Dimensions

The dimensions of the BUTTON object are determined in function of its caption or the size of possible bitmaps creating the button. In order to determine its size, all displayable texts or bitmaps are to be previewed. The object size determined in such a manner is a minimum one and may be enlarged using the mouse. Trying to minimize the object below its minimum size will fail.

Selection of buttons can be made using the mouse or the Tab (Shift + Tab) combination. Sending of values to controlled variables takes place in the following cases:

- under Send immediate - at the moment of releasing the left mouse button or after striking Enter key;
- under Send with acknowledge - after pressing the Grey+ key or by carrying out an appropriate action (Enter can be used to change the state of the SWITCH object);
- under Send with repeating - a cyclical sending is made every second until the left mouse button is pressed. Releasing the button causes sending additionally a special control word („End of repeating"). When using Enter, click it twice. The first clicking causes „pressing" of the button and cyclical sending of controlled variables. The second clicking serves to finish cyclical sending and sends a special word to the controlled variable („End of repeating").

REMARK *In case of "Send with acknowledge" or "Send with repeating" modes to quit use of buttons, click the right mouse button or Esc key.*

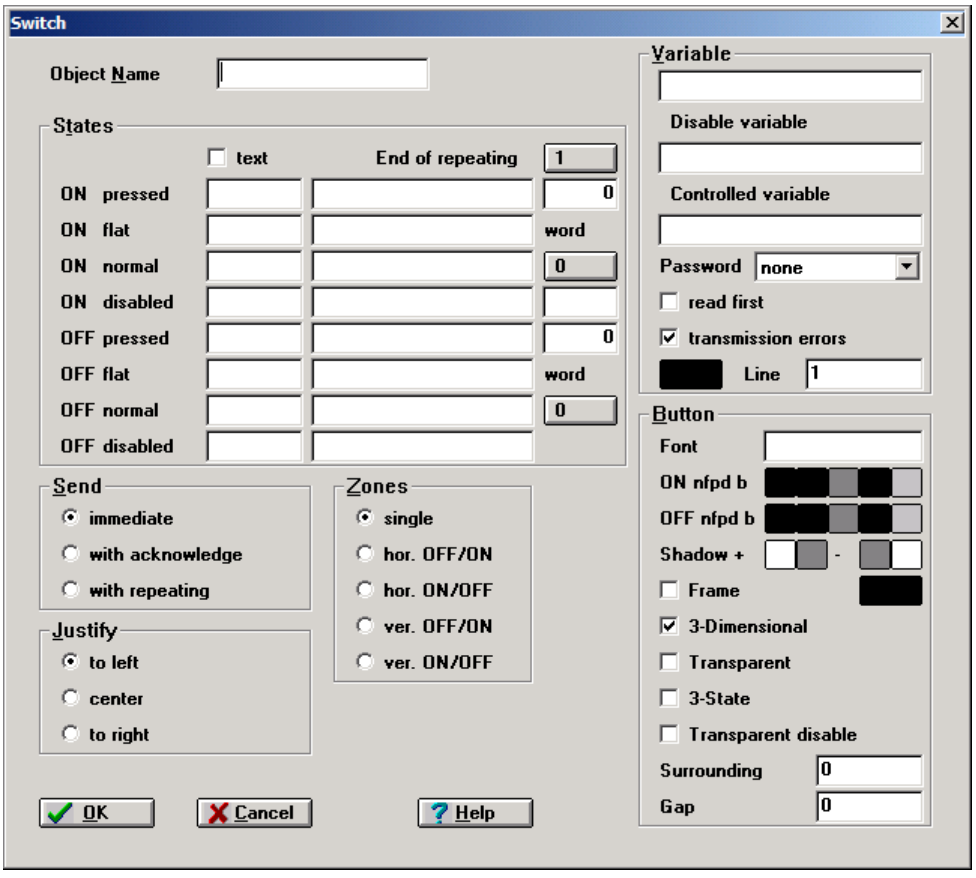


Figure 18-61. The SWITCH Object Parameterization Window.

Parameters

Object Name - this text box is used for to put in the optional name of the object. If the name of the object will not be defined, its type together with coordinates will appear on the list of objects.

States- Text **-** a check box to specify whether the buttons are to be text-described or bitmap-created. The captions and the bitmaps are placed in the boxes beneath the ***States-text*** box. REMARK: activating this field causes previously entered bitmap names to be automatically deleted.

States- ON pressed **-** three text boxes to set parameters of a type **ON** pressed button. The first box contains the caption (for example „ON"). The second one may contain the bitmap name displayed inside the button provided that the ***States-text*** box is not active. The third box contains a number in hexadecimal code, which indicates bits to be controlled in monitored variable (from *Variable* field).

Currently, there are three states for state definition in the SWITCH object:

1. entering a number, in hexadecimal code, indicating bits which are to be controlled in monitored variable;
2. entering a number in hexadecimal code with preceding = character;
3. entering a number in hexadecimal code with preceding & character; conformity of all 1 bits is required.

E.g. if the number 2 is entered to the field - then description or bitmap defined for that state will be displayed on the button (or ON-type button will be activated for double button version), if the second bit from the right, of the monitored variable, has value 1. Then clicking the button will cause performing an operation defined for it.

States- ON flat **-** two text fields allowing parameterization of inactive **Z** type button. First field contains caption (e.g.„ON"). The second field may contain the name of bitmap displayed inside the button, if the ***States-text*** field is not set. Inactive button will be flat, if the ***tri-state*** option is selected.

States- ON normal **-** two text fields allowing parameterization of unpressed **Z** type button. The first field contains caption (e.g. „ON"). The second field may contain the name of bitmap displayed inside the button, if the ***States-text*** field is not set. The third field contains the button, for which the operation is defined, which is

performed on pressing the button, under condition , that the monitored variable has the bits indicated in ***Z pressed*** line set. For example, if there is number 3 entered in this field, the operation defined for the described button will be performed, in case the monitored variable has two lowest bits set.

States- ON disabled

- three text boxes to set parameters of a OFF type disabled button. The first box contains the caption (for example „DIS"), provided that the ***transparent disable*** box is not active. The second one may contain the bitmap name displayed inside the button, provided that the ***States-text*** or ***transparent disable*** box is not active. The third box contains a number in hexadecimal code, which indicates the bits to be controlled in ***disable variable***.

Currently, there are three states for state definition in the SWITCH object:

1. entering a number, in hexadecimal code, indicating bits which are to be controlled in ***disable variable***;
2. entering a number in hexadecimal code with preceding = character; consistence of 0 and 1 bytes is required for variable value and the mask.
3. entering a number in hexadecimal code with preceding & character; consistence of all 1 bits is required.

E.g. if the number 2 is entered to the field and the second bit from right of ***disable variable*** has a value of 1- then a description or bitmap defined for that state will be displayed on the button (or ON- and OFF-type buttons will be disabled for double button version).

States- OFF pressed

- three text boxes to set parameters of a OFF type pressed button. The first box contains the caption (for example „OFF"). The second one may contain the bitmap name displayed inside the button provided that the ***States-text*** box is not active. The third box contains a number in hexadecimal code, which indicates bits to be controlled in monitored variable (from *Variable* field).

Currently, there are three states for state definition in the SWITCH object:

1. entering a number, in hexadecimal code, indicating bits which are to be controlled in monitored variable;

2. entering a number in hexadecimal code with preceding = character;
3. entering a number in hexadecimal code with preceding & character; conformity of all 1 bits is required.

E.g. if the number 2 is entered to the field - then description or bitmap defined for that state will be displayed on the button (or OFF-type button will be activated for double button version), if the second bit from the right, of the monitored variable, has value 1. Then clicking the button will cause performing an operation defined for it.

States- OFF flat **-** two text fields allowing parameterization of inactive **W** type button. First field contains caption (e.g.„OFF"). The second field may contain the name of bitmap displayed inside the button, if the ***States-text*** field is not set. Inactive button will be flat, if the ***tri-state*** option is selected.

States- OFF normal **-** two text fields allowing parameterization of unpressed **W** type button. The first field contains caption (e.g. „OFF"). The second field may contain the name of bitmap displayed inside the button, if the ***States-text*** field is not set. The third field contains the button, for which the operation is defined, which is performed on pressing the button, under condition , that the monitored variable has the bits indicated in ***Z pressed*** line set. For example, if there is number 3 entered in this field, the operation defined for the described button will be performed, in case the monitored variable has two lowest bits set.

Button-Font **-** a text box to specify the font of the button description characters. The default font is Dialog. Clicking the right mouse button on this box causes activation of the font selection window.

Button-ON **-** color boxes to set the color of the caption for the ON button (left box) for the ON button as pressed (right box) and for the background (middle button). After selection is made, the window of color selection is displayed.

Button-OFF **-** color boxes to set the color of the caption for the OFF button (left box) for the OFF button as pressed (right box) and for the background (middle button). After selection is made, the window of color selection is displayed.

Button-Shadow + **-** color boxes to specify the illumination feature (left box) and the color of the shadow of the released button (right box). Illumination is related to the button upper and left sides whilst the shadow is related to the lower right

sides. After selection is made, the window of color selection is displayed.

Button-Shadow- **-** color boxes to specify the illumination feature (right box) and the color of the shadow (left box) of the button-as-pressed. Illumination is related to the button right and bottom sides while the shadow is related to the left and upper sides.

Button-Frame **-** a check box and color box to specify whether the button should be provided with a frame and to specify the frame color.

Button-3-Dimensional **-** a check box which permits drawing of so called three-dimensional buttons.

Button-Transparent **-** a check box to draw transparent buttons. In this case, the proper buttons should be created by means of other objects.

Button-3-State **-** a check box allows third flat state of the object.

Button-Surrounding **-** a numerical box to specify the surrounding of the button in terms of pixels (Default = 0). The surrounding permits to enlarge the zone sensitive on selection (essential for small buttons).

Button-Gap **-** a numerical box to specify the gap between the buttons in terms of pixels (Default = 0). It is essential for a double button version.

Zones- single **-** a radio button to select a single button mode of operation, description of the button being changed (e.g. ON or OFF) in function of the available activity (e.g. if the drive is actually ON then it can be switched OFF and the OFF description will appear on the button).

Zones-hor. OFF/ON **-** a radio button to select the horizontally divided button with the upper part corresponding to ON state, and the bottom part corresponding to OFF state. Always one button is inactive and the second one waits for mouse click dependent of controlled device state.

Zones-hor. ON/OFF **-** a radio button to select the horizontally divided button with the opposite positions of the ON and OFF parts as described above. Always one button is inactive and the second one waits for mouse click dependent of controlled device state.

Zones-ver. OFF/ON **-** a radio button to select the vertically divided button with the left part corresponding to the OFF state and the right part corresponding to the ON state. Always one button is inactive and the second one waits for mouse click dependent of controlled device state.

Zones-ver. ON/OFF **-** a radio button to select the vertically divided button with the opposite position of the OFF and ON parts as described above. Always one button is inactive and the second one waits for mouse click dependent of controlled device state.

Variable - a text box to specify the name of the monitored variable. Specifying the name is obligatory. The state of this variable decides on the button description when displayed (ON or OFF type) - in a single button version) and on fact, which button will be active (ON or OFF type) - in a double button version respectively. Clicking the right mouse button on this box opens a window containing the list of the available variables to make selection.

Controlled Variable - a text box allowing specifying the name of controlled variable. Specifying this name is necessary. To this variable controls, declared with proper button fields, are sent. By pressing the right mouse button in this field, a window containing the list of available variables can be opened and you can make a choice in it. It is possible to specify the name of controlled variable using notation with **#** character. If in the field of variable name the #_ characters with suffix are entered, then during the application's run it will be treated as the variable with name containing the name of monitored variable instead of the # character.

Password - a combo box of the password that protects sending of the controls (four levels of password can be selected, no password is the default setting).

Read First **-** a check box to force reading before sending of the controlled variable. In such case, there is written to the controlled variable a logic sum of the sent value and of the given variable value as read from the controller with the reset bit group which has been set by the object.

Transmission Errors **-** a check box to warn about transmission errors (the object is crossed). Default option.

Line - a color box of the crossing line signalling transmission error occurrence. A numerical box to decide on line width. Selection of the color box causes displaying of the color choice window.

End of repeating - a button box (a button with a variable description) to specify either a numerical value (hexadecimal) which will be sent to the controlled variable at the end of the „With repeating" sending or the action which will be carried out in such a situation. After pressing this button, a window appears on the screen beneath to specify either the value or the action.

Send Immediate - a radio button allowing declaration of immediate value sending, after selecting the button by clicking the mouse in the object's area, in diagram refresh mode. The value, sent by object is the value corresponding to the button (OFF or ON)

Send with Acknowledge - a radio button to declare sending of the value after selecting the button by clicking the mouse button on the object box being under the diagram refresh mode and then by acknowledging. Acknowledge can be made by either clicking the button *Grey +* or executing appropriate actions. The value to be sent by the object is a value which corresponds to the button (OFF or ON).

Send with Repeating - a radio button to declare a cyclical sending (at one second period) of the value selected after the left mouse button is pressed on the object box under the diagram refresh mode. Such cyclical sending functions until the left mouse button is pressed. The value to be sent corresponds to the ON button. At the moment of release of the left mouse button the value is sent which is declared in the ***End of repeating box***.

Justify-to left - a radio button to force left alignment of the button area caption.

Justify-center - a radio button to force centering of the button area caption.

Justify-to right - a radio button to force right alignment of the button area caption.

18.4.33. SWITCH SET object

Dimensions
Object Parameters
- Object Name
- Button
- Bit Mask
- Text List
- Coding
- Outlook
- Variable
- State
- Sending

The SWITCH SET object is a set of buttons, which are commonly handled. These buttons may be either dependent each other (i.e. pressing one button causes activation of the remaining ones) or independent. This is a selectable object. Clicking on any button can cause sending the value to the controlled variable. It can be made either at the moment of button clicking or after acknowledge. Sending the variable may be password-protected. The object can also be a dynamic object. In such case, the state of the switch set buttons can be optionally set from the controller. The SWITCH SET can be a transparent object. Then the individual buttons should be created, for example using either objects of type BUTTON which do not carry out any action or objects of type PICTURES, TEXTS which do not send value to the controlled variables. It is possible to disable displaying of the so called „selection frame" which appears after selection of an object.

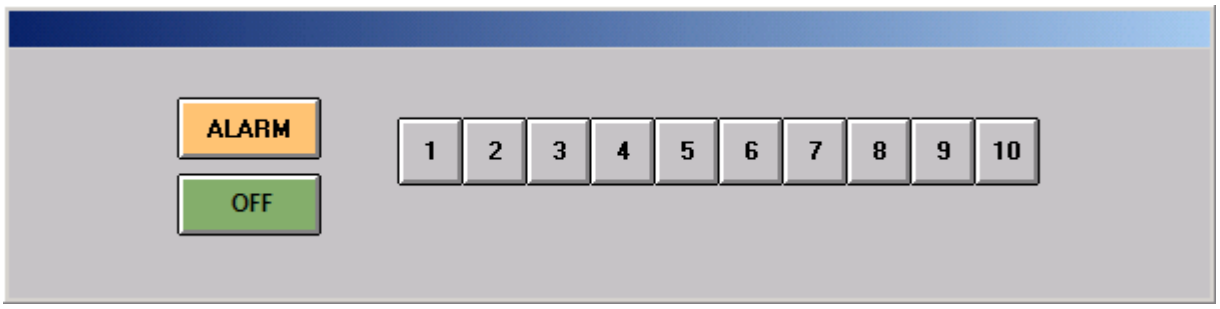


Figure 18-62. The SWITCH SET Example.

Dimensions

The dimensions of the SWITCH SET object are determined with use of mouse. After each modification of parameter settings the object dimensions are to be possibly modified because it does not resize automatically if the button descriptions are changed.

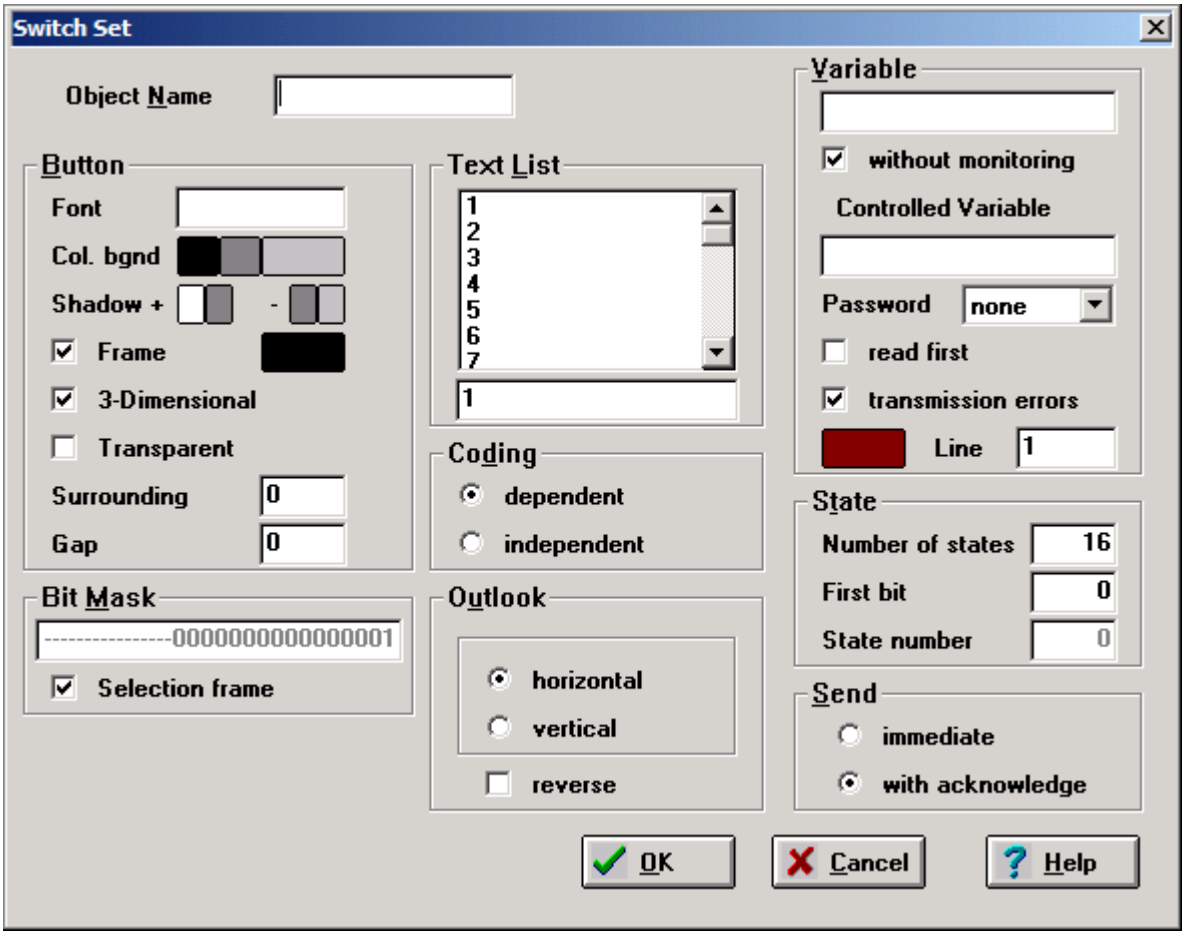


Figure 18-63. The SWITCH SET Object Parameters Window.

The button description text is to be inserted into the list position pointing with the mouse cursor. By default, the object has two states (buttons) only. The buttons may be arranged either horizontally or vertically. The buttons can be either dependent or independent. The normal assignment of the bits to the buttons means the assignment of the LSB of the word to the left button of the horizontal assignment (for the vertical outlook - the LSB corresponds to the upper button). Such arrangement can be reversed. Transmission errors are signaled by crossing the switch set with a line of the given color & thickness.

The object may be selected during refresh operation using either the mouse cursor or the *Tab* button (*Shift*+*Tab*). After the selection is made, choose the required switch set state with the mouse cursor or the *Enter* button (multiple pressing). Sending of the state is done automatically after selection/acknowledge is made. To acknowledge, press either the button *Grey+* or carry out appropriate actions. If you don't want to do selection, click either *Esc* or the right mouse button to release all switches set buttons.

For every button, individual choice of colors is possible (background, shadow, illumination etc.).

Parameters

Object Name - this text box is used enter into it the optional name of the object. If the name of the object will not be defined, its type together with coordinates will appear on the list of objects.

Button-Font **-** a text box provided to specify the font of the button description characters. The default font is Dialog. Clicking the right mouse button on this box causes activation of the font selection.

Button-Bgnd **-** color boxes provided to specify the button description color (left box), the text color for the button-released (middle box) and the background (right box). After selection is made, the window of color selection is displayed.

Button-Shadow + **-** color boxes provided to specify the illumination feature (left box) and the color of the shadow of the button-pressed (right box). Illumination is related to the button upper and left sides whilst the shadow is related to the lower right sides. After selection is made, the window of color selection is displayed.

Button-Shadow - **-** color boxes provided to specify the illumination feature (left box) and the color of the shadow of the button-as-pressed (right box). Illumination is related to the button upper and left sides whilst the shadow is related to the lower and right sides. After selection is made, the window of color selection is displayed.

Button-Frame **-** a check box and a color box provided to specify whether the button should be provided with a frame and to specify the frame color.

Button-3-Dimensional **-** a check box that permits drawing of so called three-dimensional buttons.

Button-Transparent **-** a check box to draw transparent buttons. In this case, the proper buttons should be created by means of other objects.

Button-Surrounding **-** a numerical box provided to specify the surrounding of the button in terms of pixels (Default = 0). The surrounding permits to enlarge the zone sensitive on selection (essential for small buttons).

Button-Gap **-** a numerical box provided to specify the gap between buttons in terms of pixels (Default = 0).

Bit Mask - a numerical box provided to display the monitored variable (if the ***without monitoring*** option has not been used) and the variable, which can be sent from the object corresponding to this state.

Selection Frame - a check box permitting additional displaying the so called „Selection Frame" (Object contour) after the object is selected during the diagram refresh. The frame helps to know quickly which object is the last selected one.

Text List - list of text as superimposed on the buttons, corresponding to the individual states of the object. To change text, select it on the list, insert a new text in the box below and press *Enter*. Trying to declare a text for a non-existing state will fail.

Coding-dependent - a radio button provided to choose a dependent object coding and handling. This method of coding presumes the possibility of pressing of only one button at a time (as in the radio-set band switch) and of sending only one 1 at a time.

Coding-independent - a radio button provided to choose an independent object coding and handling. This method of coding presumes the possibility of sending of a group of ones (contemporaneous pressing of more buttons).

Outlook-horizontal **-** a radio button provided to arrange the buttons horizontally.

Outlook-vertical **-** a radio button provided to arrange the buttons vertically.

Outlook-reverse **-** a check box provided to reverse the bit assignment in the word to the individual buttons. (The normal assignment of the bits to the buttons means the assignment of the LSB of the word to the left button of the horizontal outlook whilst for the vertical outlook - the LSB corresponds to the upper button).

Variable - a text box provided to specify the name of the monitored variable. Specifying the name is not necessary if the option ***without monitoring*** has been selected. Clicking the right mouse button on this box opens a window containing the list of the available variables to make selection.

Controlled Variable - a text box that allows specifying the name of controlled data. Specifying this name is necessary. By pressing the right mouse button in this field a window with list of available variables can be displayed, and you can make a choice in it. It is possible to specify the name of status data, using notation with **#** character. If in the field of variable name the #_ characters with suffix are entered, then during the application's run it will be treated as the variable with name containing the name of monitored variable instead of the # character.

Without Monitoring **-** a check box to set the object parameters in such manner that the displayed state is not dependent on any system variable and serves only to indicate the value to be sent.

Password - password Combo box that protects sending the controls (four levels of password can be selected, no password is the default setting).

Read first - a check box provided to force reading values before sending the controlled variable. In such case, there is written to the controlled variable a logic sum of the sent value and of the given variable value as read from the controller with the reset bit group which has been set by the object

Transmission errors **-** a check box provided to warn about transmission errors (the object is crossed). Default option.

Line - a color box of the crossing line to signal transmission error. A numerical box provided to decide on line thickness. Selection of the color box causes displaying of the color selection window.

State - Number of States - a numerical box provided to declare the number of states of the object. By default, the object has 16 states (i.e. 16 buttons). Maximum number of states is 32. The values associated with the states (buttons) correspond to the successive bits of the word.

State - First Bit - a numerical box provided to declare the first (least significant) bit of the monitored variable, the value of such variable is displayed as a state of the successive keyboard keys. The contents of this box permit for an unambiguous determination of the position of the group of the bits, which control the states displayed. The first bit corresponds to the LSB whilst the group size corresponds to the number of states.

State - State Number - a numerical box provided to display the number of the state selected. Selection is made by clicking on the appropriate item on the ***Text List***.

Send Immediate - a radio button to declare immediate sending of the value, after the proper button is selected, by clicking the mouse button on the

area of the button box while being under the diagram refresh mode. This type of sending is reasonable if the ***Coding-dependent*** mode has been selected otherwise pressing the successive buttons (for setting the keyboard) will cause sending of a series of values. The object sends a value corresponding to the keyboard state.

Send with Acknowledge - a radio button to declare sending of the value after the proper buttons are selected by clicking (many times) the mouse button on the area of the button boxes while being under the diagram refresh mode and then by acknowledging. Acknowledge of sending can be made by either clicking the button "*Grey+*" or executing appropriate actions. This type of sending is reasonable if the object is under ***Coding-independent*** mode although may also be used in another case. The object sends a value corresponding to the keyboard state.

18.4.34. SYNCHRONIZER Object

Dimensions
Object Parameters
Object Name
Controlled Variable
Value

SYNCHRONIZER is an invisible object. It enables a value to be written to the controlled variable. The write operation is synchronized by the "PERFORM_INPUT, all" action.

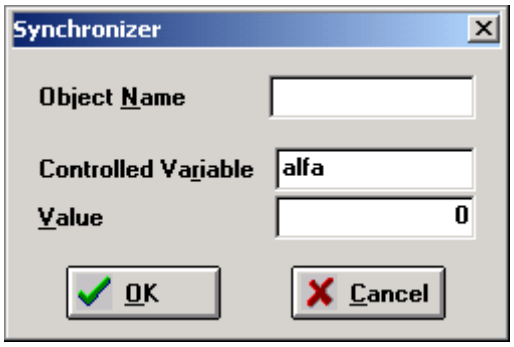


Figure 18-64. The SYNCHRONIZER Object Parameters Window.

Dimensions

The dimensions of the object SYNCHRONIZER are of no concern (the object is invisible).

Thanks to usage of the SYNCHRONIZER , the controller program can be optimized. Setting a defined variable is a message to the controller that the operator is working with a specified mask. Thus, only a limited set of controlled variables can be sent and there is no longer necessary to verify all the controlled variables in the controller.

Parameters

Object Name - this text box is used for to put in the optional name of the object. If the name of the object will not be defined, its type together with coordinates will appear on the list of objects.

Controlled Variable - a text box provided to specify a name of the controlled variable. Specifying the name is obligatory. By clicking the right mouse button on this box you can open a window containing the list of available variables to make selection.

Value - a text box to insert a hexadecimal value (default = 0).

18.4.35. TEXT Object

Dimensions
Object Parameters
- Object Name
- Text
- Source of Text
- VarDef
- Attributes
- Justify
- View
- Docking
- Parameters

Configuring from the VarDef

The TEXT object enables a text to be displayed on the diagram. This is a static object. It is possible to define its font and colors. The text can be inserted in another object (for example PICTURE) without clearing of the background. Optionally, it is possible to use a text identifier, which is inserted in the application configuration file in ***Text parameters*** parameter with use of Architect: Architect > *Fields and Computers* > *Masks* module > *Text parameters* tab.

This approach permits the use of the same diagram for different applications (which vary as to their descriptive parts). It is possible to write in a vertical layout (downward, without reversing of the characters). A non-standard application of the TEXT object is to use it to display the diagram name (precisely - the file containing this diagram) after the box "insert diagram name" is set. Copying such arranged object to another diagram cause the new diagram name to be displayed by the object. The TEXT object, which displays a diagram name ignores the contents of the "text" box.

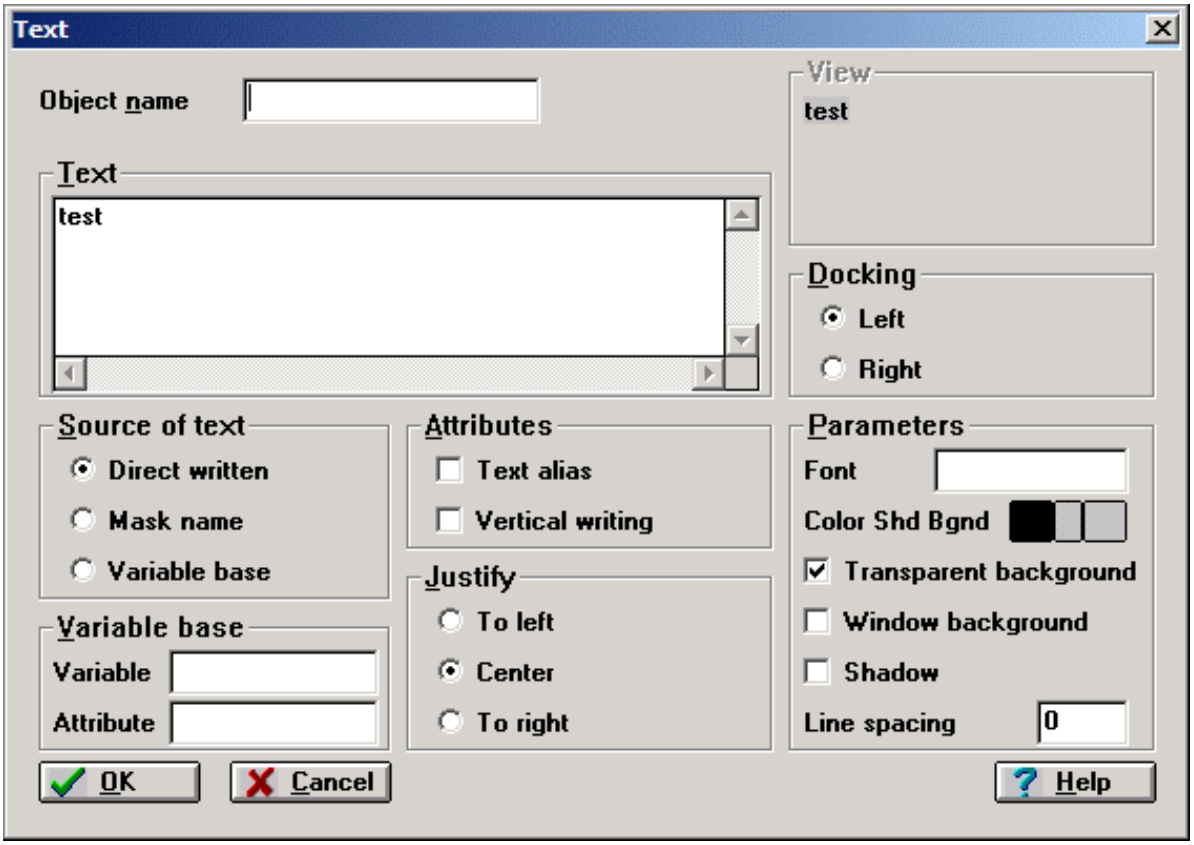


Figure 18-65. The TEXT Object Parameters Window.

Dimensions

Both the height and the width depend on the text and on the font selected.

When receiving an empty text, the height is not changed.

The text can be written in (and edited) directly over the diagram without need of opening of the dialog box.

Parameters

Object Name - this text box is designed to enter the optional name of the object. If the name of the object will not be defined, its type together with coordinates will appear on the list of objects.

Text - one-or-multiple-line text field containing text entered in the object's field. This text can be entered directly on the diagram. Maximum number of characters is 1000. Moving to the new line is made after pressing *Shift+ENTER*. *Backspace* key is only supported in the last line of entered text.

View - displays declared text, also with shading.

Docking **-** a radio button declaring the point of docking the text on diagram. In case of the change of size of displayed text in the base it will be widen to the right, if it's docked on the left side. Docking on the right side causes widening to the left.

Source of text - direct written **-** a radio button that enables declaring the text by the operator.

Source of text - diagram name **-** a radio button allowing to declare, that the diagram name will be entered instead of text, on which the given object is located. It makes the later editing and searching for diagrams easier. It is only active for one line text.

Source of text - Variable Database **-** a radio button enabling outputting the value of any attribute located in the variables base.

Variable Base **-** a text box enables inserting the variable name and the name of displayed variable's attribute.

Attributes - ***Text alias*** - a check box provided to inform that in the text list there are only the names of text equivalents, whilst the real texts are situated in the application *.xml file in ***Text parameters*** parameter (declared with use of Architect program > *Fields and Computers* > *Masks* module > *Text parameters* tab).

Attributes - Vertical writing - a check box also enables declaring vertical inserting the text(from up to down without rotating the characters). It is only active for the one-line text.

Justify - radio buttons forcing aligning the string in object's area to the left, centering it or aligning to the right.

Parameters

- Font **-** a text box allowing entering the name of font for displaying the text. Not specifying the font causes accepting default font (Dialog). Pressing the right mouse button in this field causes moving to the window of font selection.

- Color - a color box enables setting the font color after opening the window of color selection.

- Shd - a color box, enables setting the font shading (if this option has been selected) after opening the window of color selection.

- Bgnd **-** a color box, enables setting the text background color after opening the window of color selection.

- Clear background - a check box that allows clearing the text background
(otherwise the text would be written on existing background, which could also be useful).

- Window background - a check box that allows displaying the text background in background color set for the diagram.

- Transparent background - a check box that allows displaying a transparent background.

- Shadowing - a check box which allows to force displaying shaded font.

Line spacing - field for declaring the text line spacing (important for multiline text only). By default is set to **0,** which means standard spacing for selected font. Entering a positive value in this field causes increasing the spacing by specified number of pixels, and entering a negative value causes decreasing the spacing.

Configuring from the Variable Definitions Database

There is also a possibility of introducing the value of any attribute contained in the VarDef.

The ***Base*** field enables entering the variable and attribute name. By pressing the right mouse button in the ***name*** field you can be moved to the variable selection window. In the ***attribute*** field the attribute name should be entered. It can be any of existing attributes. In typical applications use of text attributes, belonging to ***Name***, ***Description*** or ***Unit*** type is expected. But these may also be numeric attributes, like e.g. ***ConversionFunctionRangeFrom*** and then the contents will automatically be converted to the text form.

For the object to display the attribute value from the base, in the selection field ***Text source*** from the VarDef item should be selected. Text from Variable Database may be also written vertically. The TEXT object displaying the contents of database field ignores the contents of "text" field.

Example window of definition of the TEXT object, displaying Description from database field for the A000_P2 variable- see: The TEXT Object Parameters Window.

18.4.36. TEXTS Object

Dimensions
Object Parameters
- Object Name
- Text
- State
- Bit Mask
- Sending
- Text List
- Coding
- Vertical Writing
- Parameters
- Justify
- Variable

Example of Coding
Configuring from the Variable Definitions Database

TEXTS object enables one of many (17) specified texts to be displayed over the diagram. Texts may be displayed vertically. This is a dynamic object (if it is a multistate object). The text selected for displaying depends on the value of the monitored variable. Optionally, the TEXTS object may be also a selectable object and may serve to send a specified value to the control variable. Sending of the value can be password-protected. Instead of texts it is possible to use text identifiers, which are declared in the application configuration file in ***Text parameters*** parameter with use of Architect: Architect > *Fields and Computers* > *Masks* module > *Text parameters* tab.

This approach permits the use of the same diagram for different applications (which vary as to their descriptive parts).

Figure 18-66. The TEXTS Examples.

Dimensions

The dimensions of the TEXTS object depend on the text and on the font selected. In order to determine the size, previewing is made of all texts displayable in the text window. Such determined object size is a minimum size, which may be enlarged using the mouse. Trying to reduce the object below its minimum size will fail.

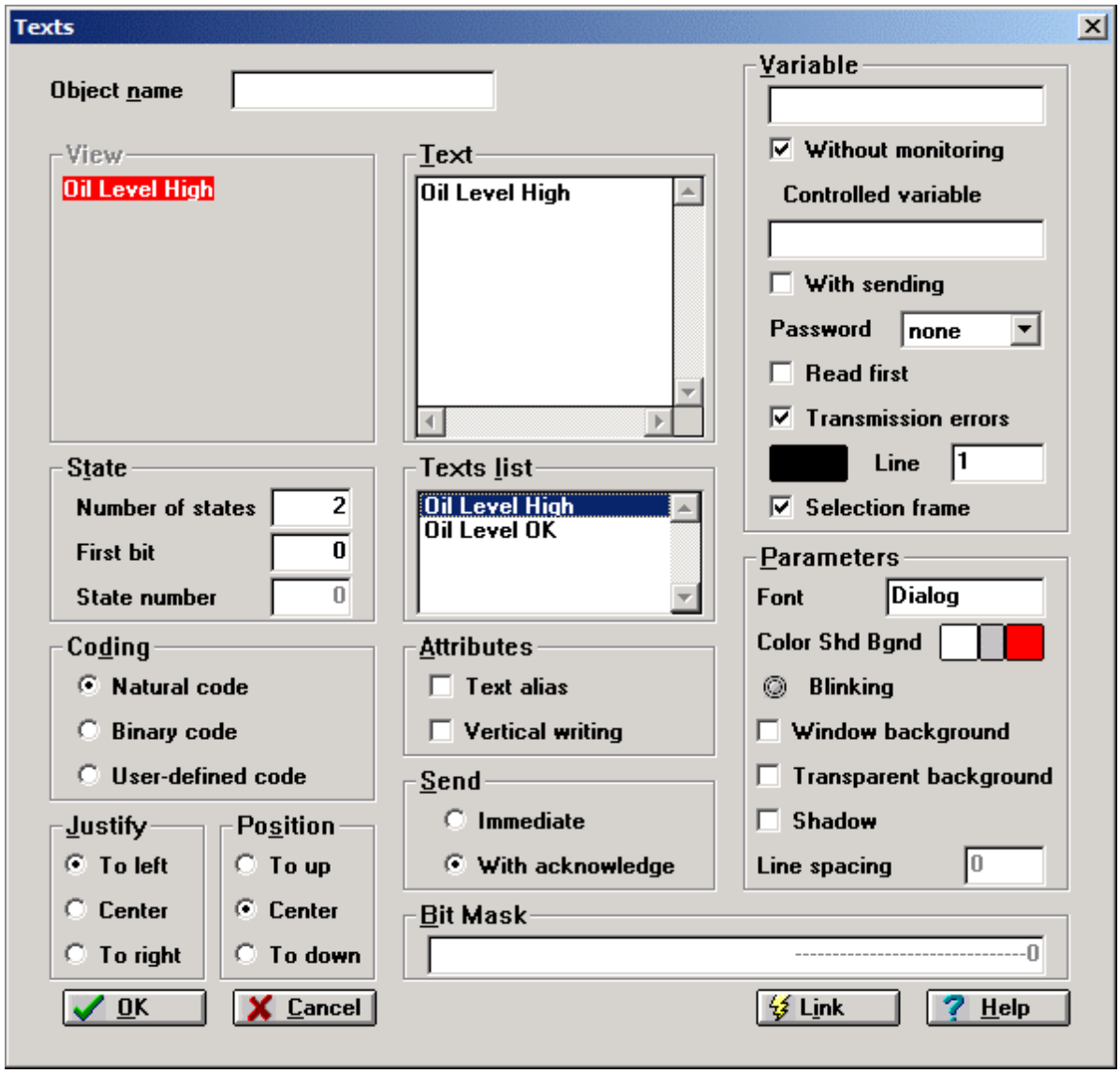


Figure 18-67. The TEXTS Object Parameters Window.

Parameters

Object Name - this text box is designed to enter the optional name of the object. If the name of the object will not be defined, its type together with coordinates will appear on the list of objects.

Text - a text box containing a text of the selected state.

State - Number of States - a numeric box that enables declaring a number of states of the object. By default the object has 1 state and then it is the static object. Maximum number of states is 33 for natural coding, 64 for binary coding and user-defined coding. The individual states correspond with values that are result of the chosen coding mode.

State - First Bit - a numerical box provided to declare the first (least significant) bit of the monitored variable,

the value of such variable is displayed as one of many declared texts. The contents of this box (valid for both natural and binary coding only) permit for an unambiguous determination of the position of the group of the bits which control the texts as displayed in the word. The first bit corresponds to the LSB whilst the group size corresponds to the coding method.

State - State Number - a numerical box only provided to display the number of the state selected. Selection is made by clicking on the appropriate item of the ***Text List***.

Bit Mask - a numerical box provided to display the monitored variable (and the variable at the same time, which has been sent from the object being set as ***With control***) corresponding to the given state. Under user-defined coding, it is possible to set the diagram by writing either hexadecimal or binary code corresponding to the given state with possibly marking of the bits such coded with the „-" symbol.

The ***'Coder'*** window is used to define the states in user-defined coding mode:

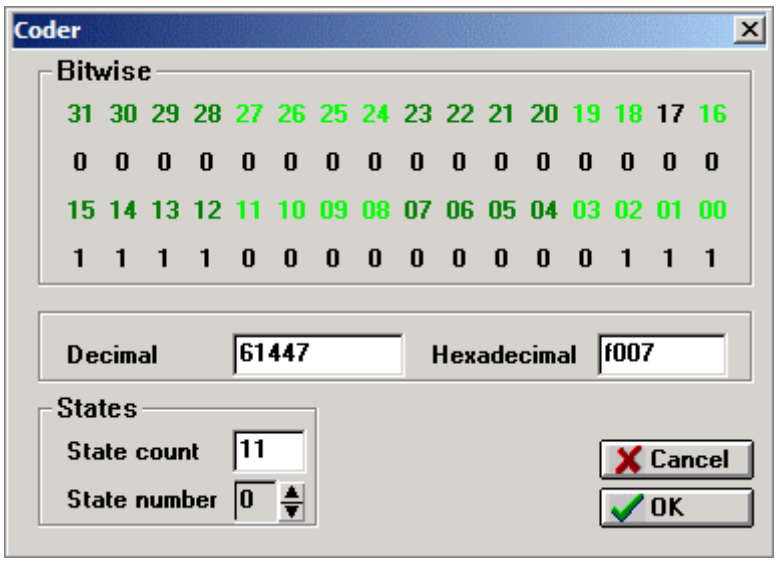


Figure 18-68. The 'Coder' Window.

The number of states recognized by the object is passed in *State count* field. This value may be also set in the main window for object defining. The state the code of which should be defined is selected in *State number* field. While setting the number in *Bitwise* frame, the current state coding is displayed - to change it, one should click with mouse on a bit value. Successive click operations cause change of the value in the following sequence:

- – bit value is unimportant
1 – bit has to be set

0 – bit has to be set to zero

Sate coding is automatically stored in the moment of choosing the other state.

The *Decimal* and *Hexadecimal* fields display decimal and hexadecimal value of defined state.

Selection Frame - a check box that permits additional displaying of the so called *Selection Frame* (object contour) after the object is selected during the diagram refresh. The frame helps to know quickly which object is the last selected one.

Send Immediate - a radio button provided to declare immediate sending of the value, after the corresponding text is selected, by clicking the mouse button on the object box while being under the diagram refresh mode (only possible if the object is set ***With control***). This type of sending is reasonable if the object has two states otherwise the object state previewing as resulted from the selection of the successive states will cause sending of a series of values. The value by object is the number that corresponds to the state number of object.

Send with Acknowledge - a radio button provided to declare sending of the value after the corresponding text is selected by clicking (many times) the mouse button on the object box while being under the diagram refresh mode (only possible if the object is set ***With control***) and then by acknowledging. Send acknowledgement can be made by, either clicking the button "*Grey+*" or executing appropriate actions. This type of sending is reasonable if the object is a multistate object. The value by object is the number that corresponds to the state number of object.

Text List - a list of texts corresponding to the individual states of the object. To change text, select it on the list, insert a new text in the ***Text*** box and press *Enter*. Trying to declare a text for a non-existing state will fail.

Text alias - a check box provided to inform that in the text list there are only the names of text equivalents, whilst the real texts are situated in the application *.xml file in ***Text parameters*** parameter (declared with use of Architect program > *Fields and Computers* > *Masks* module > *Text parameters* tab).

Vertical writing - a check box provided to declare that the text should be written vertically (downwards, with no character reversing).

Coding - Natural Code - a radio button provided to choose a natural code (for coding methods - see chapter after Parameter List).

Coding - Binary Code - a radio button provided to choose a binary code (for coding methods - see chapter after Parameter List).

Coding - User-Defined Code **-** a radio button provided to choose an user-defined code (for coding methods - see chapter after Parameter List).

Parameters-Font **-** a text box provided to specify the font for displaying the text corresponding to the given state. It is possible to set individually this box for each text. The default font is Dialog. Clicking the right mouse button on this box causes activation of the font selection window.

Parameters –Color Shd Bgnd **-** a color field, which allows setting the text color (left field), shading (if declared) and text background color (right field) corresponding to the given status. After selecting this field a color selection window is displayed. It is possible to set this field individually for each text. Indicator to the right from the color fields enables setting the blinking attribute.

Parameters-Window Background **-** a check box provided to force setting of the text background color same as the window background color. It is possible to set individually this box for each text.

Parameters-Transparent Background **-** a check box provided to set a transparent background.

Shadowing **-** a check box which allows to declare text shading in accordance with color defined in ***Parameters-Color/Shadow /Background*** position.

Line spacing **-** declaration of line spacing in multiline text.

Justify-to left - a radio button to force left alignment of the object descriptive text. It is possible to set individually this box for each text.

Justify-center - a radio button provided to force centering of the object descriptive text. It is possible to set individually this box for each text.

Justify-to right - a radio button provided to force right alignment of the object descriptive text. It is possible to set individually this box for each text.

Position – to up - a radio button for selecting vertical justification to the upper edge of object's area. Individual justification for each of the statuses (texts), doesn't depend on writing direction.

Position – center - a radio button for selecting vertical justification with vertical centering in object's area. Individual justification for each of the statuses (texts), doesn't depend on writing direction.

Position – to down - a radio button for selecting vertical justification to the lower edge of object's area. Individual justification for each of the statuses (texts), doesn't depend on writing direction.

Variable - a text box provided to specify the name of the monitored variable. Specifying the name is not obligatory if the ***Without monitoring***

option has been selected. Clicking the right mouse button on this box opens a window containing the list of the available variables to make selection.

Controlled Variable - a text box allowing specification the name of controlled variable. Specifying it is obligatory, if the ***With control*** option has been selected. By pressing the right mouse button in this field, a window containing the list of available variables can be opened and you can make a choice in it. It is possible to specify the name of controlled variable using notation with **#** character. If in the field of variable name the #_ characters with suffix are entered, then during the application's run it will be treated as the variable with name containing the name of monitored variable instead of the # character.

Without Monitoring **-** a check box provided to set the object parameters in such manner that the displayed state would be independent on any system variable but serve only to indicate the value to be sent.

With Sending **-** a check box provided to set the object parameters in such manner that the value corresponding to its state would be sent to the system variable. Sending of the value takes place after the object is selected, for example by clicking on it with the mouse - the action, which will follow depends on the send mode.

Password - a password combo box that protects sending of the controls (four levels of password can be selected, no password is the default setting).

Read First **-** a check box provided to force reading before sending of the controlled variable. In such case, there is written to the controlled variable a logic sum of the sent value and of the given variable value as read from the controller with the reset bit group which has been set by the object.

Transmission errors **-** a check box to warn about transmission errors (the object is crossed). Default option.

Line - a color box of the crossing line signalling transmission error occurrence; a numerical box to decide on line width. Selection of the color box causes displaying of the color choice window.

The text is written in the list by pointing the proper place with the mouse cursor. Pointing the text in the list causes displaying of it in a special window of the dialog box. The text display attributes (font, alignment and colors) will be assigned to the text pointed in the list.

By default, the object has one state only i.e. it is a static object. Crossing the object with horizontal lines of specified color and thickness signals transmission errors.

During the refresh operation, the object can be selected using either the mouse cursor or the *Tab* key (*Shift*+*Tab*). After the selection is made, choose the given object state with the mouse cursor or the *Enter* key. Sending of the state is done automatically after selection/acknowledge is made. To acknowledge, press the *Grey+* key or carry out appropriate actions. To quit the selection, click either *Esc* or the right mouse button.

There exist the possibility exists to use the conversion function, the values of it are the indexes of successive texts as well as to use the another conversion function which changes the object state number to the sent value (when the option *With control* is active). The decoding of a fixed-point monitored variable value (and coding of possibly sent values as well) using one of three object-embedded strategies is possible too.

Natural coding

The group of bits is checked. Number of bits is equal to the *number of states*, but number of the LSB defines "offset". The group of bits can consist only one value of "1". Number of the state is calculated as position of the "1" within the group of bits (the LSB position is number one). The zero state corresponds with the group of all zeros.

Binary Coding

The group of bits is checked. Number of bits is equal to the binary logarithm of *number of states*, but number of the LSB defines "offset". Number of state is calculated as content of the tested bit group.

User-Defined Coding

The whole word is checked. The "offset" parameter is ignored. The user assigns a string of ones and zeros (the *don't carry* states can be masked by "-") to each state. Number of state is calculated as the first of states, that the user assigned the same bits on the appropriate positions (regardless to the masked bits).

For the objects which are set *With control*, the option *Read first* may be used. In such case, there is written to the controlled variable a logic sum of the sent value and of the given variable value as read from the controller with the reset bit group which has been set by the object.

EXAMPLE

The object has four states and the binary coding has been chosen. To code four binary states (00,01,10,11) a two-bit group in the word is necessary. The first bit of the group determines the contents of the box ***State-First bit***. If it equals for example to 4, then the state of only the fifth and sixth bits will affect the object state. The remaining bits have no influence on displaying of the object what is illustrated below.

```
----------xx----
```

("-" denotes a non-affecting bit, "x" denotes a bit which affects the object state)

If you define four texts e.g. "1", "2", "3"and "4" (consecutively) which correspond to these four states, the "1" text corresponds to the state "00", the "2" text corresponds to the state "01" and so on. With writing, if you choose the "4" text, the corresponding state "11" of the fifth and sixth bits i.e. the decimal number 48 is sent to the variable. When the ***Read first*** box is activated, this mechanism is somewhat different - at first the given variable is read and then the fifth and sixth bits are reset, the state corresponding to the number of the text selected being overwritten to these bits.

```
1000000000100000    / the read variable
1000000000-—0000    / the read variable with reset bits „-"
----------11----    / bits which are set by the object
1000000000110000    / the sent variable
```

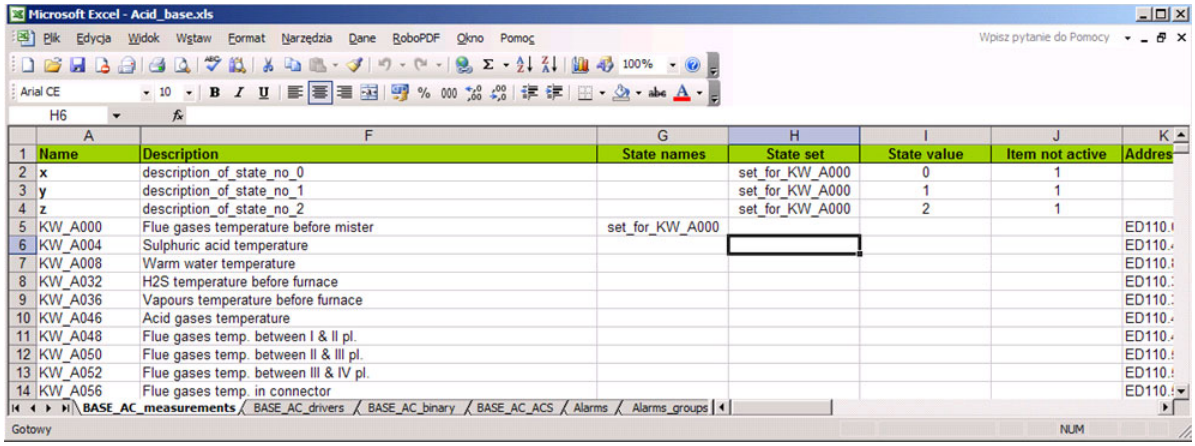
Configuring from the Variable Definitions Database

All possible texts may be declared in a database of variable definitions, one variable (database row) per text of one state. For each such variable the database must contain some text to be displayed (the *Description* attribute), as well as the following attributes:

State name - name of collection of states /or collection of descriptions for variable states;
State set – name of that collection of states, to which the given database row belongs
State value – decimally expressed bit mask value for the state described in the given database row
Item not active – flag indicating that the variable is not visible for Asmen and Aspad.

All variables with declarations of texts to be displayed must be associated with monitored variable by means of a common name of collection of its states (specified in the *State set* column of all those variables). That same name must be entered as the *State names* attribute of the monitored variable.

Descriptions of states of the monitored variable are displayed on technological masks according to variable states.



	A	F	G	H	I	J	K
1	Name	Description	State names	State set	State value	Item not active	Addres
2	x	description_of_state_no_0		set_for_KW_A000	0	1	
3	y	description_of_state_no_1		set_for_KW_A000	1	1	
4	z	description_of_state_no_2		set_for_KW_A000	2	1	
5	KW_A000	Flue gases temperature before mister	set_for_KW_A000				ED110.
6	KW_A004	Sulphuric acid temperature					ED110.
7	KW_A008	Warm water temperature					ED110.
8	KW_A032	H2S temperature before furnace					ED110.
9	KW_A036	Vapours temperature before furnace					ED110.
10	KW_A046	Acid gases temperature					ED110.
11	KW_A048	Flue gases temp. between I & II pl.					ED110.
12	KW_A050	Flue gases temp. between II & III pl.					ED110.
13	KW_A052	Flue gases temp. between III & IV pl.					ED110.
14	KW_A056	Flue gases temp. in connector					ED110.

Figure 18-69.Declaration of Texts for Monitored Variable States in Variable Definitions Database.

To load texts for variable states in TEXTS configuration window, you should enter in the *Variable* field the name fo monitored variable, and then push the *Link* button.

18.4.37. WORK POINT Object

Dimensions
Object Parameters
- Object Name
- Range of X
- Range of Y
- Background
- Trace
- Variables

Configuring from the VarDef

The WORK POINT object enables displaying the cursor, moving over previously prepared background (in the form of bitmap tables). The bitmap, designed as background for displaying the status of X and Y variables has no size limits. It can be placed in a separate file in BMP format, on the disk, in the directory declared as diagram's directory. Then, in the object configuring window only the filename with BMP extension should be given.

There is also the possibility of declaring the object **without** the bitmap background. Current status of X and Y variables is then displayed directly on the mask– you can previously prepare a static background of WORK POINT using LINE HV, RECTANGLE, PICTURE or similar objects.

Cursor may leave a trail. It is possible to use standard cursor or declare any cursor, pointing the values of current variables. It can be any bitmap located in **asix** bitmaps folder.

Signalling the communication and measurements errors is possible.

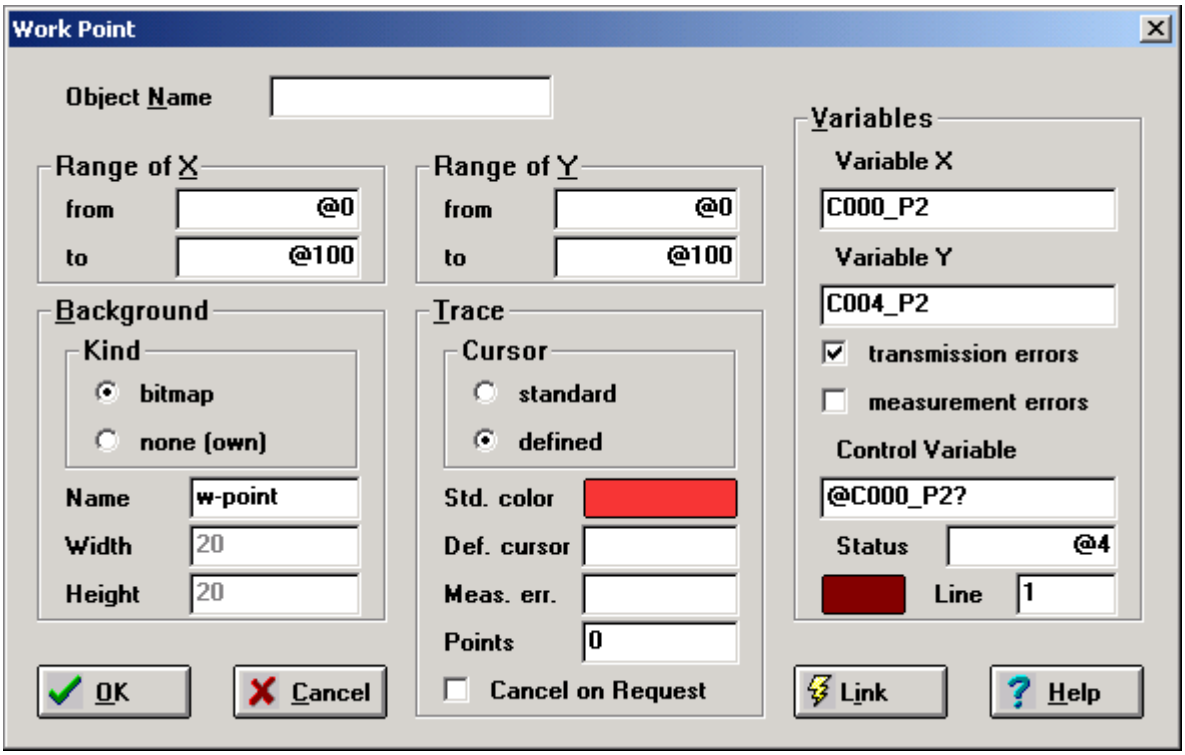


Figure 18-70. The WORK POINT Object Parameters Window.

Dimensions

Dimensions of WORK POINT object are strictly specified, if the bitmaps were used as a graphic background. In other case the dimensions can be set freely, with the mouse or by specifying directly in the **Background** space of the object configuring dialog window.

Parameters

Object Name - this text box is used to enter into it the optional name of the object. If the name of the object will not be defined, its type together with coordinates will appear on the list of objects.

Variables-Variable X - a text box provided to specify the name of the measurement variable X (corresponding to the horizontal axis). It is obligatory to specify this name. By clicking the right mouse button on this box you may open the list of available variables and make selection.

Variables-Variable Y - a text box provided to specify the name of the measuring variable Y (corresponding to the vertical axis). It is obligatory to specify this name. By clicking the right mouse button on this box you may open the list of available variables and make selection.

Range of X-from - a numerical box provided to insert a numerical value defining the lower range of the variable X. Default = 0.

Range of X-to - a numerical box provided to insert a numerical value defining the upper range of the variable X. Default = 100.

Range of Y-from - a numerical box provided to insert a numerical value defining the lower range of the variable Y. Default = 0.

Range of Y-to - a numerical provided box provided to insert a numerical value defining the upper range of the variable Y. Default = 100.

Background-Name **-** a text box provided to specify the name of a bitmap (being the cursor background). The bitmap name can be either written in the dialog box or selected by clicking the right mouse button. After selection is made, a window will appear for previewing and selecting the bitmaps belonging to the **asix** system bitmap pool.

Background-Kind-bitmap - a radio button designed for bitmap selection for object background displaying.

Background-Kind-none(own) - a radio button used for declaring the background without the bitmap - the object is then transparent and all objects, previously put on the mask in the place where the *Work Point* object is located, are visible.

Background-Width - a nemrical box designed for object width specification in case of selecting the ***Background-type-none*** **(*custom*)** field. Value changes automatically after scaling the object with a mouse.

Background-Height - a numerical box for object height specification in case of selecting the ***Background-type-none*** **(*custom*)** field. Value changes automatically after scaling the object with mouse.

Trace-Cursor-standard - a radio button provided to declare the use of standard cursor.

Trace-Cursor-defined - a radio button provided to declare the use of non-standard cursor. The cursor's shape is defined by the bitmap declared in ***Trail-cursor def*.**

Trace-Std. color - a color box provided to set the cursor color.

Trace-Def. cursor - the field for selecting the shape of defined cursor, displayed during object run, if measurements errors do not occur.

Trace-Meas. err. - the field for selecting the shape of cursor displayed in case of measurement errors occurrence in the measuring variables.

Trace-points - a numerical box to set the number of points of the trace left behind the cursor (Default = 0 - no trace).

Trace-Cancel on Request - a check box to cancel the trace (by clicking within the object area).

Transmission errors **-** a check box provided to signal the transmission errors. The cursor changes its

color if a transmission error occurs. It is a default option.

Measurement errors - a check box provided to signal the measurement errors. A line surrounds the cursor if a measurement error occurs. It requires specifying the ***Control Variable***.

Control Variable - a text box provided to specify the name of a control variable. If the ***Measurement errors*** option is selected, specification of this name is obligatory. Clicking the right mouse button on this box enables opening of the window of available variables and making selection.

Status - a text box provided to insert a hexadecimal mask. If the product of this mask and the control variable is different from zero then a measurement error occurs.

Line - a color box defining the cursor and the line, which surrounds the cursor when errors occur and a numerical box, which serve to determine the line thickness. Selection of the color box causes displaying of the color selection window.

Configuring from the Variable Definitions Database

The *WORK POINT* object is provided with features enabling setting the following parameters from the VarDef:

- range of variations of both input variables e.g. 0-200, 100-300,
- name and mask of control data, placed in the base for the X variable.

This object has ***Link*** button, pressing of which causes inserting @ character, and next the contents of suitable field in the database. The Link button works under condition, that field data in object definition window is empty.

18.4.38. CHART Object

DESCRIPTION REGION
CURVE REGION
LEGEND REGION

CHART object enables a graphic presentation to be made of archived data collected by means of the ASPAD program. The data can be presented under the form of linear diagrams and bar graphs oriented horizontally or vertically. Every CHART is automatically scaled relatively to the region declared by the designer. The chart may comprise many curves, the number of which is only limited by the color palette, line thickness and patterns available. The curves can represent both current values and archived values of the measurement variables as well as pattern trends. Additionally the pattern trend has the possibility of locating the trend anchor in any orientation towards the OX axis.

If the mask with the trend containing a pattern variable (see: *18.4.38.6. Setting Parameters*) is buffered, the pattern anchor will be stored. The archive curves can be shifted in respect to the current values. For bar graphs, it is possible to alter their attributes (width, height and color) in function of the value of the measurement variable associated.

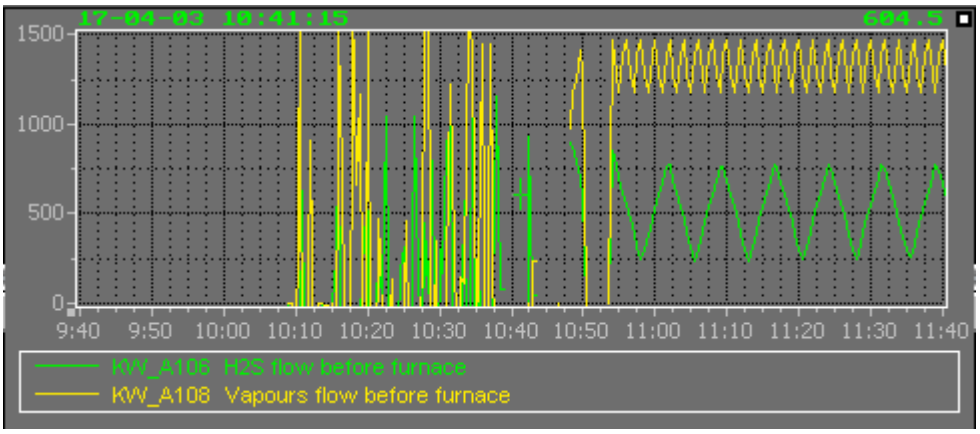


Figure 18-71. The CHART Object Example.

Each chart occupies a rectangle place on the screen, which consists of the following components:

DESCRIPTION REGION - includes the elements that are used to read and interpret the data included in the curve region,

CURVE REGION - includes the curves being a graphic presentation of the measurement variables,

LEGEND REGION - includes information on the curves and on measurement variables as well.

The **DESCRIPTION REGION** incorporates the following elements:

selection marker - designed to indicate an active chart,

cursor date - presents the date corresponding to the position of the cursor on the time axis (if the cursor is ON) under the DD-MM-RR format; the character color matches the color of the curve selected on the chart,

cursor time - presents the time corresponding to the cursor position (if it is ON) on the time axis under the HH:MM:SS format; the character color matches the color of the curve selected on the chart,

cursor value - presents a value corresponding to the cursor position (if it is ON) on the time axis; the character color matches the color of the curve selected,

chart title - describes the contents of the curve region; normally it includes a name, description and measure unit of the process variable,

description of time axis - presents numerical values which make graduation of the time axis and determine the

range of the curves displayed. The axis description is divided in subranges described with the markers to help interpretation of the displayed curves. The format of the axis marker description can be: HH:MM:SS, HH:MM or HH,

description of value axis - presents numerical values which make graduation of the time axis and determine the range of the curves displayed. The axis description is divided in subranges described with the markers to help interpretation of the displayed curves. The format of the axis marker description is either a decimal number with its fraction or an exponential number where the exponent is situated over the value description,

exponent - a box containing a power (mantissa) of the values presented in the exponential notation of the axis. The exponential notation is $\mathbf{c*10^m}$ where c is a decimal number called a characteristic of a logarithm whilst m is a mantissa. For example, 2300 in exponential notation may be written as $2.3*10^3$,

axis marker - a graphic mark (vertical or horizontal dash of different length) which divides the time axis and the value axis in equal subranges to help interpretation of the displayed curves. These marks may be of two types: described (longer) and non described (shorter),

history marker - a graphic mark (rectangle) which appears at the moment when the current curve does not have within its runtime a point corresponding to the present time,

border frame - a frame to separate the descriptive region from the curve region.

The **CURVE REGION** incorporates the following elements:

current curves - curves made of measurement points connected with straight lines to illustrate current changes of values of the process variables. In order to distinguish them, different colors, widths and patterns of the drawing line are used. Because time is the argument for the variables and the curves serve to show changes currently then the curve offset behind the curve region provokes also the offset of the curve region contents along with the time axis description. If the curves do not contain a point corresponding to the present time, the offset does not take place (a history marker is visible).

archive curves - curves made of measurement points connected with straight lines to illustrate archived changes of values of the process variables. In order to distinguish them, different colors, widths and patterns of the

drawing line are used. No offset of the curve region contents with elapsed time takes place. The curves may be shifted in respect to the current curves by means of special functions of the chart service.

pattern curves - curves defined with Pedit program, (detailed description in Pedit.hlp. file) With the time passing they don't cause shifting the contents of curves area. They can be shifted towards the current curves with special curve handling functions, the same, which are used with archival curves.

cursor - a graphic mark (crossed lines) which serves to point any place within the curve region. Its color depends on the color of the curve as selected from the chart. It is used to:

- inform on which curve is currently selected,
- read the point coordinates within the curve region,
- establish a reference point for the range modification of the time axis and the value axis.

read grid - horizontal or vertical lines beginning from the markers of the time axis description and running through the whole curve region. They make easier interpretation and read of positions of points over the curves.

The **LEGEND REGION** incorporates the following elements:

line pattern - a graphic presentation of the lines used to draw the chart curves.

variable name - a measured variable name as a reference name for creating the chart curves.

description of variable - description of a measured variable displayed on the basis of data written in the asix system configuration files.

All the a/m elements are displayed with colors valid for curves defined in the CHART object.

18.4.38.1 Chart Service Functions

Mouse operations shown below describe the way of selecting CHART objects.

- After pressing the left mouse button in CHART object area follows:
  - unselecting other CHART objects;
  - selecting indicated object;
  - activating (making it current) the object.
- After pressing the left mouse button in the area of object different than CHART follows:
  - making the indicated object current;
  - keeping the CHART objects selection.

- After pressing the left mouse button + Ctrl key in the area of CHART object follows:
  - selecting indicated CHART object;
  - keeping the other CHART objects selection within the limits of mask.
- After pressing the left mouse button in the area of CHART object, the context menu is displayed, enabling performing operations on selected CHART objects.

Described change of selection of CHART – class objects allows performing functions on the group of selected CHART objects. From the operational point of view it allows gaining the following characteristics:
- functions activated from the keyboard for current CHART object will be multiplied for selected CHART objects;
- the toolbar of CHART object, activated upon user's demand, controls the group of selected CHART objects;
- context menu initiates performing of some functions on the group of CHART objects.

18.4.38.2 Context Menu

The context menu of the CHART object is activated by pressing the right mouse button in the object's area. It has the following form:

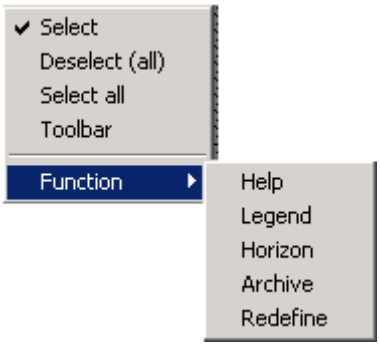


Figure 18-72. The Context-Sensitive Menu of the CHART Object.

Particular positions of menu have following meanings:

Select **-** enables turning on or off the selection of given CHART object;

Deselect (all) **-** enables unselecting all selected CHART objects;

Select all - enables selection of all CHART objects within the limits of mask;

Toolbar **-** enables showing or hiding the bar of tools controlling the functions of selected CHART objects; this operation can also be performed with double-clicking in the area of CHART object;

Function **-** shows sub-menu containing the functions of CHART object, that can be performed from this location; sub-menu contains the functions, the performance of which activates dialog windows; here belong functions:
Help,
Legend,
Horizon,

Archive,
Redefine.

18.4.38.3 CHART Toolbar

In order to make handling of CHART object easier, it has been provided with specialized toolbar, enabling mouse support. Additionally, by using the method of selection of CHART objects, a possibility of performing operations on the group of charts is received.

Activating the toolbar is realized by double-clicking the mouse in the area of any CHART object or through the context menu. Closing the toolbar is performed by pressing the window's system button or with the object's context menu.

The toolbar has the following form:

Figure 18-73. The CHART Toolbar.

It appears as an independent window, with „always on top" feature, resizable and movable. This window contains several bars, with object's functions grouped according to their purpose. These are among other:

- functions related to cursor support;
- functions related to general chart support;
- functions related to changing the ranges on axes;
- functions related to archive handling.

The form of CHART object toolbar after expanding all functions is following:

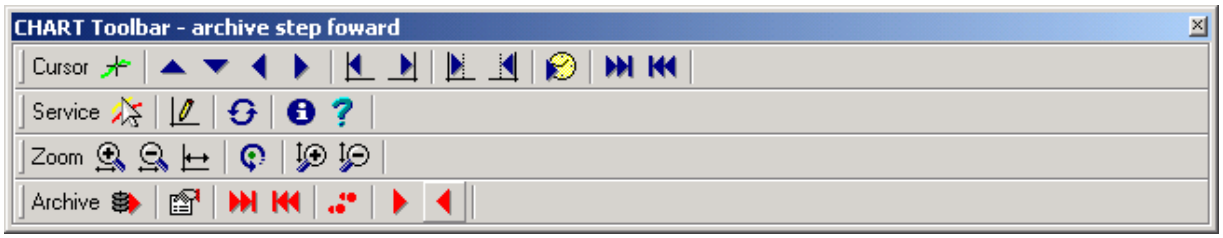


Figure 18-74. The CHART Toolbar.

Division of the functions into groups corresponds to the division proposed in object's help window invoked with *Ctrl+F1* key combination.

Functions with suitable key code and graphic representation are listed in the following tables.

Table 18-2. Functions with Suitable Key Code and Graphic Representation for the CHART Object.

No.	Function	Key Code	Graphic Representation
Cursor			
1	Turn on/off the cursor	Ins	
2	Cursor up	↑	
3	Cursor down	↓	
4	Cursor to the left	←	
5	Cursor to the right	→	
6	Cursor to the left edge	Home	
7	Cursor to the right edge	End	
8	chart beginning from the cursor	Ctrl P	
9	chart ending from the cursor	Ctrl K	
10	Return to the current time	Ctrl Home	
11	A horizon forwards	PgUp	
12	A horizon backwards	PgDn	

Table 18-3. Functions with Suitable Key Code and Graphic Representation for the CHART Object (continuation).

No.	Function	Key code	graphic representation
Service			
13	Curve selection	Center	
14	Drawing the chart	Ctrl Enter	
15	Curves redefinition	Ctrl D	
16	Curves info	Ctrl L	
17	Help window	Ctrl F1	

Table 18-4. Functions with Suitable Key Code and Graphic Representation for the CHART Object (continuation).

No.	Function	Key code	graphic representation
Zoom			
18	Increase the chart in the time axis	Ctrl ←	
19	Decrease the chart in the time axis	Ctrl →	
20	Horizon change	Ctrl H	
21	Start parameters	Ctrl Center	
22	Increase the chart in the 0Y axis	Ctrl ↓	
23	Decrease the chart in the 0Y axis	Ctrl ↑	

Table 18-5. Functions with Suitable Key Code and Graphic Representation for the CHART Object (continuation).

No.	Function	Key code	graphic representation
Archive			
24	Turn on/off the archive	Ctrl Ins	
25	Archive parameters window	Ctrl A	
26	Archive a horizon forwards	Ctrl PgUp	
27	Archive a horizon backwards	Ctrl PgDn	
28	Step: pixel or axis marker	Enter	
29	Archive a step forwards	Ctrl +	
30	Archive a step backwards	Ctrl -	

The toolbars can be freely moved within the ranges of window. Every toolbar contains suitable set of buttons. Every button corresponds to one function of the CHART object and has the picture representing this function. After pointing the button with mouse cursor a prompt is displayed, showing the function's name. Pressing the button causes performing the function on selected object or a group of CHART objects.

18.4.38.4 Keys which Activate the Chart Service Functions

Cursor Manager
Chart Service
Modification of Presentation Range
Archived curves

Operations accessible from the toolbar can be also performed with the use of keyboard. A defined set of keys with functions listed below is used in this purpose:

Cursor Manager

Ins	- switches the cursor ON/OFF within the curve region; as a result, a cruciform cursor appears with the same color as that of the curve selected. Markers appear on the time axis and value axis to make easier read of positions of the cursor.
↑	- shifts the cursor by a pixel (point) upwards.
↓	- shifts the cursor by a pixel (point) downwards
←	- shifts the cursor by a pixel (point) leftwards.
→	- shifts the cursor by a pixel (point) rightwards.
Home	- shifts the cursor on the time axis to the beginning of the curve region.
End	- shifts the cursor on the time axis to the end of the curve region.
PgUp	- scrolls the curve region one page up on the axis; the „page" is intended as a time horizon of the curve region.
PgDn	- scrolls the curve region one page back on the axis; the „page" is intended as a time horizon of the curve region.
Ctrl-Home	- returns to the actual time moment; this function is used for quick return to display of the curve, which shows the current time moments. It is particularly useful after use of ***PgUp*** or ***PgDn*** functions.
Ctrl-P	- shifts the left edge of the curve region to the time axis marker situated on the left side of the cursor (a quick establishing of the beginning of the curve region using the cursor).
Ctrl-K	- shifts the right edge of the curve region to the time axis marker situated on the right side of the cursor (a quick establishing of the beginning of the curve region using the cursor).

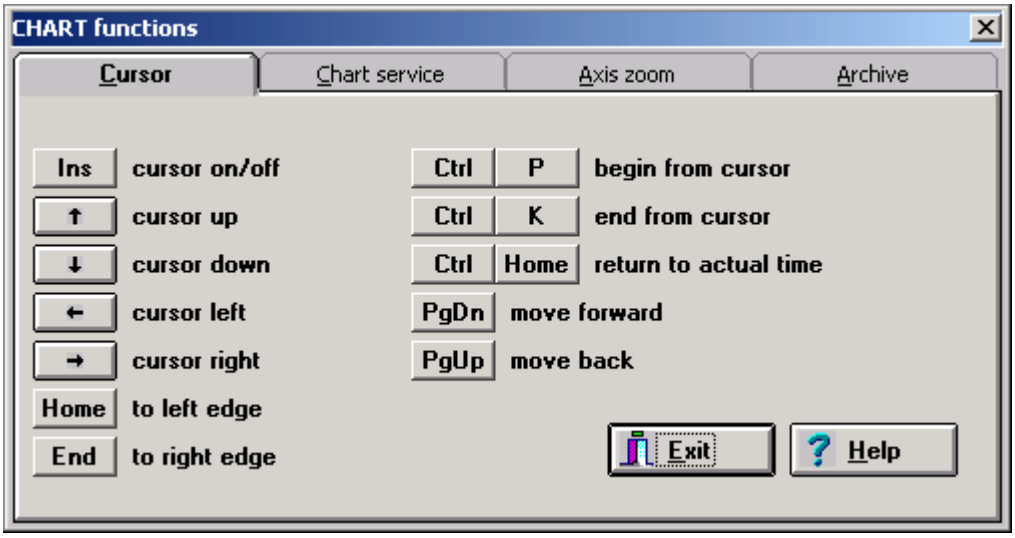


Figure 18-75. The 'CHART Functions' Window - Cursor Tab.

Chart Service

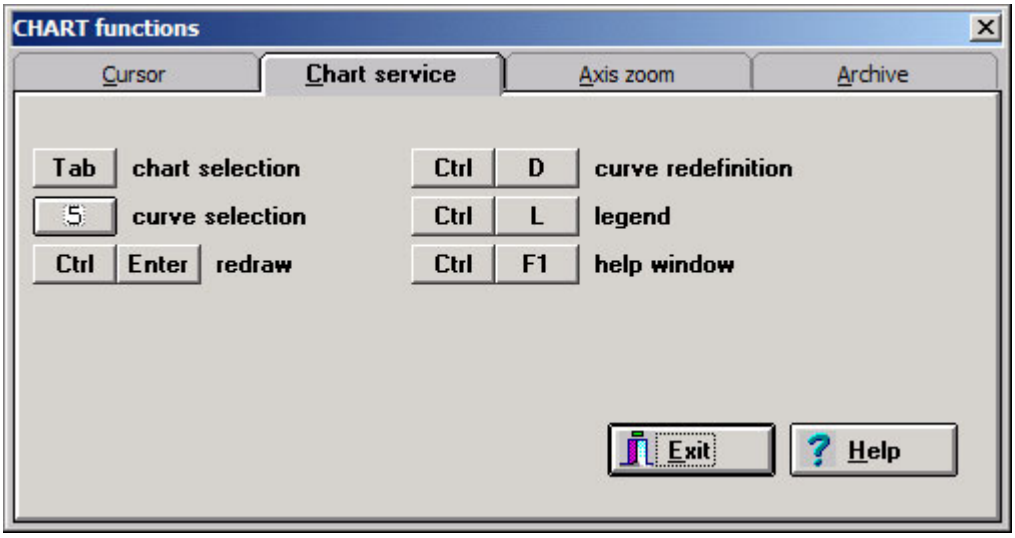


Figure 18-76. The 'CHART Functions' Window - Chart Service Tab.

Tab - selects the curve region.

Center - (key 5 of the numeric keyboard) selects a curve within the curve region.

Ctrl-Enter - redraws the curve region contents.

Ctrl-F1 - makes active a dialog window, which informs on functions of the CHART object buttons.

Ctrl-L - makes active a dialog window, which informs on the curves defined for a selected CHART object

Figure 18-77. The Chart Legend Window.

Ctrl-D - makes active a dialog window, which allows to redefine the CHART object curves. This window is available only for objects with dynamic parameters.

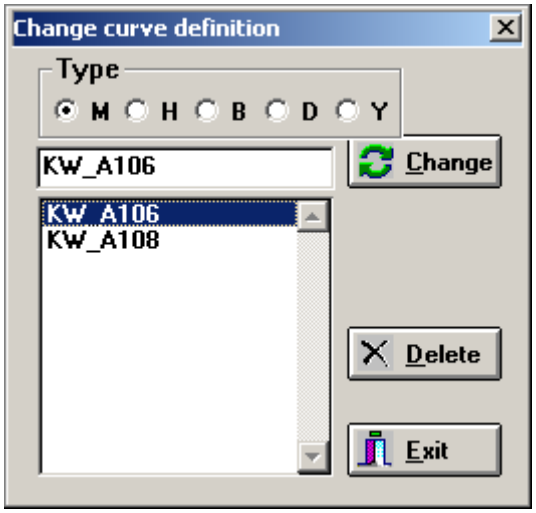


Figure 18-78. The 'Change Curve Definition' Window.

Modification of Presentation Range

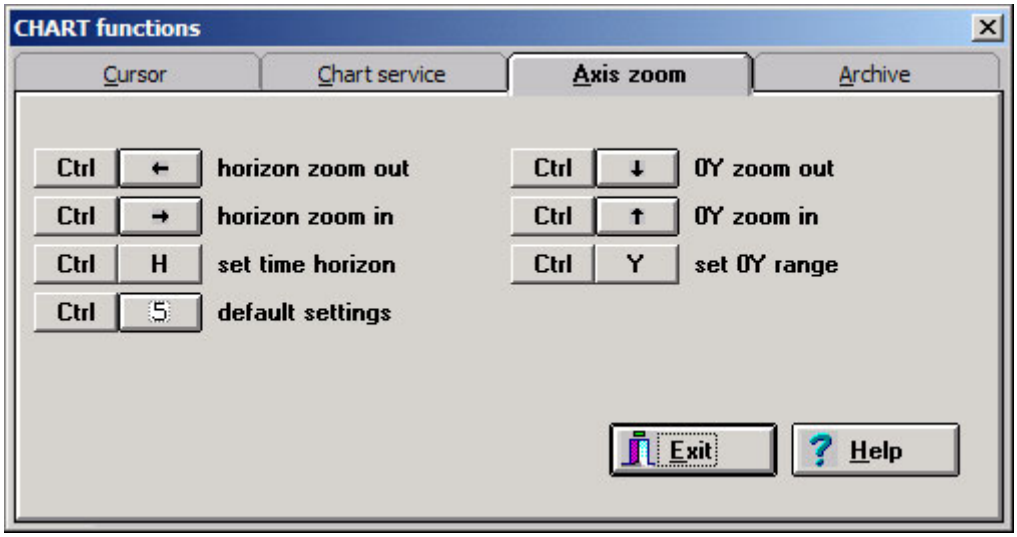


Figure 18-79. The 'CHART Functions' Window - Axis Zoom Tab.

Ctrl-→ - serves to increase the time axis precision of displayed curves; time horizon is reduced twice. If the cursor is not active, such change is accomplished in relation to the middle of the current time axis range. If the cursor is active, the change of time horizon is accomplished in relation to the cursor; after such operation is done, the cursor is positioned in the middle of the time axis range.

Ctrl-← - serves to decrease the time axis precision of displayed curves; time horizon is extended twice. If the cursor is not active, such change is accomplished in relation to the middle of the current time axis range. If the cursor is active, the change of time horizon is accomplished in relation to the cursor; after such operation is done, the cursor is positioned in the middle of the time axis range.

Ctrl-H - serves to change time horizon of displayed time curves; time horizon length is changed by determining its span on the basis of the dialog window:

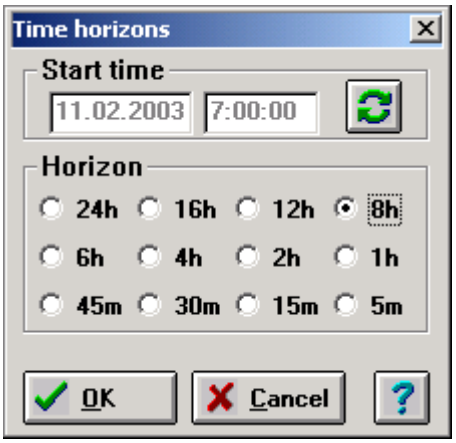


Figure 18-80. The 'Time Horizons' Window.

The boxes of the „Start time" group allows to determine the beginning of displaying of curves over the preset time horizon.

Ctrl-↑ - serves to increase the value axis precision of displayed curves; the value axis length is reduced twice. If the cursor is not active, such change is accomplished in relation to the middle of the current time axis range. If the cursor is active, the change of the axis length is accomplished in relation to the cursor; after such operation is done, the cursor is positioned in the middle of the value axis range.

Ctrl-↓ - serves to decrease the value axis precision of displayed curves; the value axis length is augmented twice. If the cursor is not active,

such change is accomplished in relation to the middle of the current time axis range. If the cursor is active, the change of the axis length is accomplished in relation to the cursor; after such operation is done, the cursor is positioned in the middle of the value axis range.

Ctrl-Center - (*Ctrl-Number5*) returns to the initial parameters of both the time horizon and the value axis range. The initial parameters are the parameters as determined by the object designer.

Archived Curves

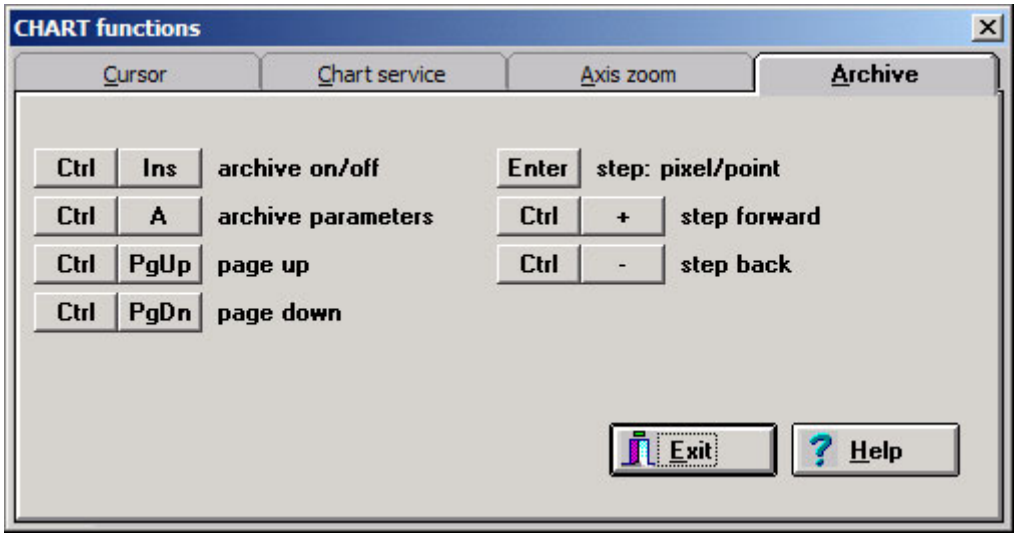


Figure 18-81. The 'CHART Functions' Window - Archive Tab.

Ctrl-Ins - switches ON/OFF the display function of the archived curves.

Ctrl-A - activates a dialog window, which permits setting parameters of the archived curves such as the date and the archive beginning.

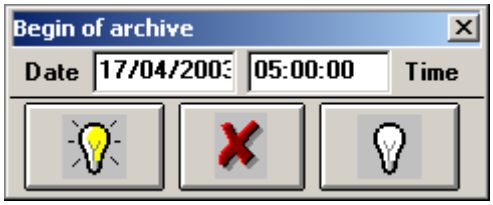


Figure 18-82. The 'Begin of Archive' Window.

Enter - sets the offset step of the archived curves being equal to 1 pixel or one time axis marker.

Ctrl-+ - serves to offset up the archived curves by one pixel or one marker in respect to the current curves.

Ctrl--	- serves to offset back the archived curves by one pixel or one marker in respect to the current curves.
Ctrl-PgUp	- serves to offset up the archived curves by one „page" in respect to the current curves .
Ctrl-PgDn	- serves to offset up the archived curves by one „page" in respect to the current curves.

18.4.38.5 Mouse-activated Chart Service Functions

selection	- serves to select a curve by positioning the mouse cursor in the curve region and clicking the left mouse button;
shift the cursor	- serves to shift the cursor in the curve region (with the cursor ON) by positioning the mouse cursor in the place in question and clicking the left mouse button;
read the value	- functions currently during shifting of the mouse cursor within of the curve region of a chart selected;
context menu	- opened after clicking the right mouse button on the chart field;
toolbar	- can be opened by quick double-clicking the mouse on the chart field.

18.4.38.6 Setting Parameters

General Parameters
OX AXIS
OY AXIS
Axes Description Parameters
Grid Parameters
Parameters - Chart Colors
Parameters - Cursor
Parameters - Curve Manager
Curve Parameters

The dialog window of CHART object parameters enables the global object parameters to be set.

General Parameters

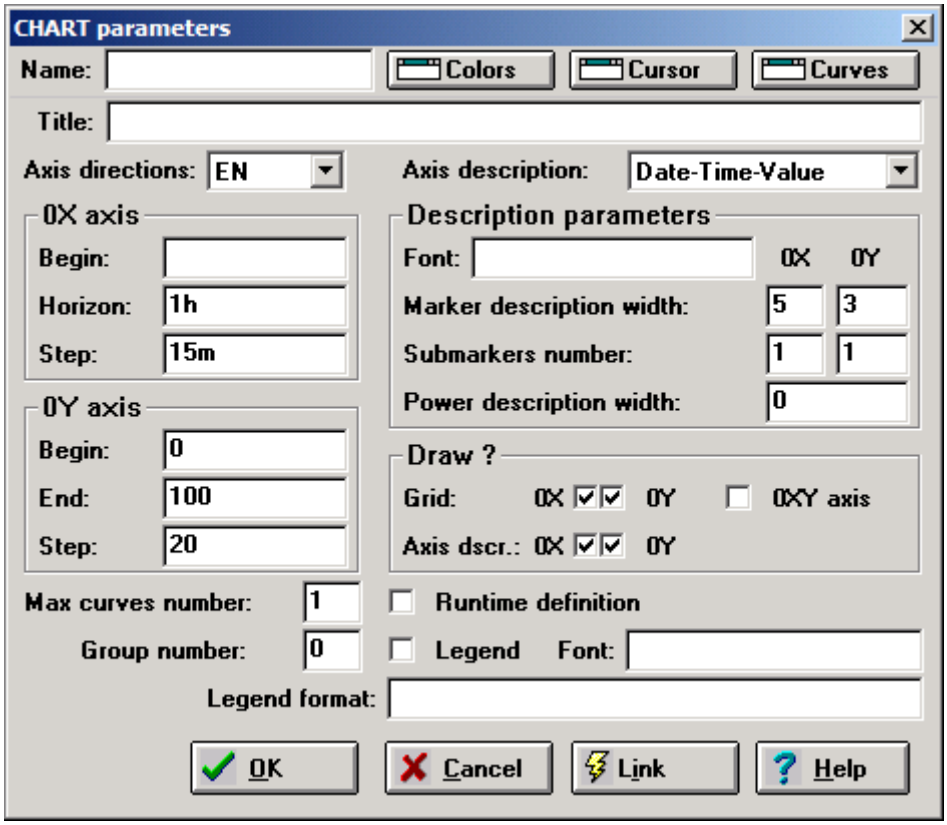


Figure 18-83. The CHART Parameters Window.

Name - to specify the object name.

Colors - a button to activate the chart color selection dialog window.

Cursor - a button to activate the dialog window, which determines the form and reactions of the chart cursor.

Curves - a button to activate the dialog window, which manages the curves within the chart.

Title - to specify the chart title describing the chart contents.

Axis directions - to specify directions of the chart axes. The options are: EN, ES, WS, WN, NE, SE, SW, NW. The first letter denotes the time axis direction, the second one denotes the value axis direction (North, East, South, West);
Default = EN.

Axis description - to specify the method of chart description; handles the following chart elements: *cursor date*, *cursor time and cursor value.* The following combinations are possible: Date-Time-Value, Date-Time or None;
Default = Date-Time-Value.

Max curves number - maximal number of curves presented on the chart; default value is 0;

Group number - allows to group CHART objects; each CHART object we want to group should have the same group number; all objects belonging to the same group are handled in the synchronous way – for example, operation of changing the rolling range is performed simultaneously on all objects of the common group;

Runtime definition - allows on-line parameterization of the chart; default value – no;

Legend - allows to display the legend on the chart; default value – no;

Font - font type used for the Legend; default font – Dialog;

Legend format - the field is used to declare the text defining the legend; parameters are being loaded from the VarDef are declared through names of parameters written in braces, e.g.:

{*Name*} -- {*Unit*}

If *Legend format* field remains empty or the base of variables isn't used, the standard format will be applied:

Name Range Description.

It is also possible to put the stamp {-} in *Legend format* to display a dash of the curve colour in the legend.

0X Axis

is time? - an option to specify if the axis is a time axis. Default = YES

Begin - to specify the initial value of the chart argument. If time is the argument this box is ignored.

Horizon - to specify the presentation horizon of the argument. If time is the argument, the values have the following format:
number[time unit of measure]
where:
time unit is one of the marks: s-seconds, m.-minutes, h-hours;
Default = 1h.

Step - to specify the argument axis scaling step. Its format is the same as for the *Horizon* box;
Default = 10m.

0Y AXIS

Begin - to specify the initial value of the axis description;

End - to specify the final value of the axis description;

Step - to specify the value axis scaling step;

Description Parameters

Font - axis description font;
Default = Dialog

Marker description width - **0X**: number of characters of the 0X axis marker description;
Default = 5
0Y: number of characters of the 0Y axis marker description;
Default = 3

Submarkers number - **0X**: number of subranges between the 0X axis markers;
Default = 1
0Y: number of subranges between the 0Y axis markers;
Default = 1

Power description width - number of characters of the power for a value under exponent notation; Default = 0

Greed Parameters

Grid - **0X**: *draw* option for the 0X axis grid;
Default = YES;
0Y: *draw* option for the 0Y axis grid;
Default = YES;

Axis dscr - **0X**: *draw* option for the 0X axis description;
Default = YES;
0Y: *draw* option for the 0Y axis description;
Default = YES;

The parameter window dialog incorporates three buttons: *Colors*, *Cursor* and *Curves*, pressing of which activates successive dialog windows.

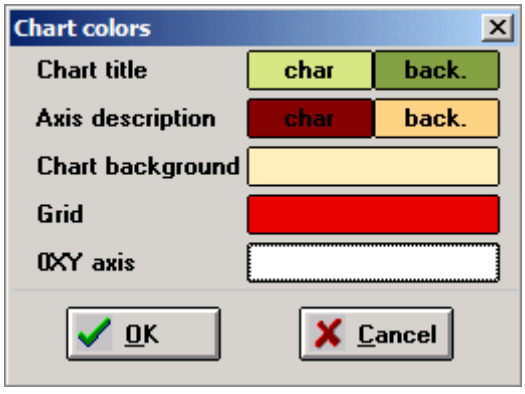


Figure 18-84. The 'Chart Colors' Window.

Parameters – Chart Colors

Chart title - a chart title characters color; Default = yellow;

	a chart title background color; Default = violet.
Axis description	**-** an axis description characters color; Default = Parisian blue; an axis description background color; Default = blue.
Chart background	**-** a curve region background color; Default = grey.
Grid	**-** a read grid color; Default = marine blue.
0XY axis	**-** a color of the co-ordinate system axes; Default = white.

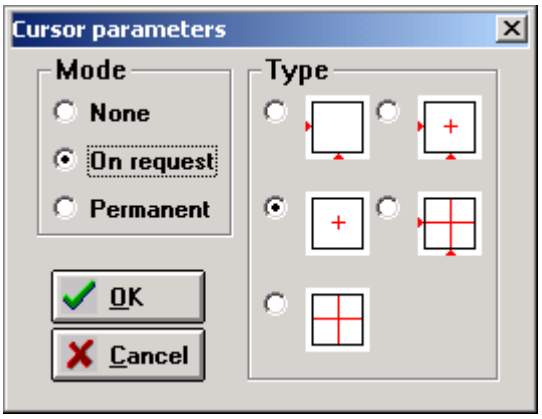


Figure 18-85. The 'Cursor Parameters' Window.

Parameters – Cursor

Mode	**-** a group of boxes to select a mode of operation of the cursor within the curve range (None, On request, Permanent); Default = On request
Type	**-** a group of boxes to select a graphic presentation of the cursor within the curve range; Default = +

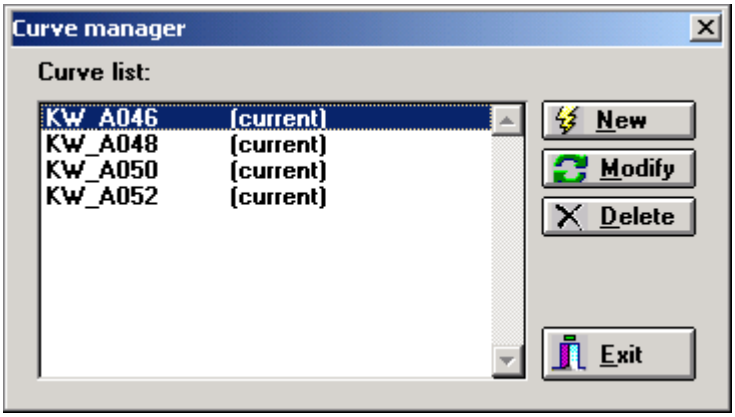


Figure 18-86. The 'Curve Manager' Window.

Parameters – Curve Manager

Definition of curves - a list of the curves as defined within the object; the description specifies variable names and types.

New - the button creates a new curve within the object (the number of curves is limited by their maximum value) and activates the *Curve Parameters* dialog window.

Modify - the button activates the *Curve Parameters* dialog window to permit modifications of parameters of the selected curve.

Delete - deletes the selected curve.

Parameters – Curve Parameters

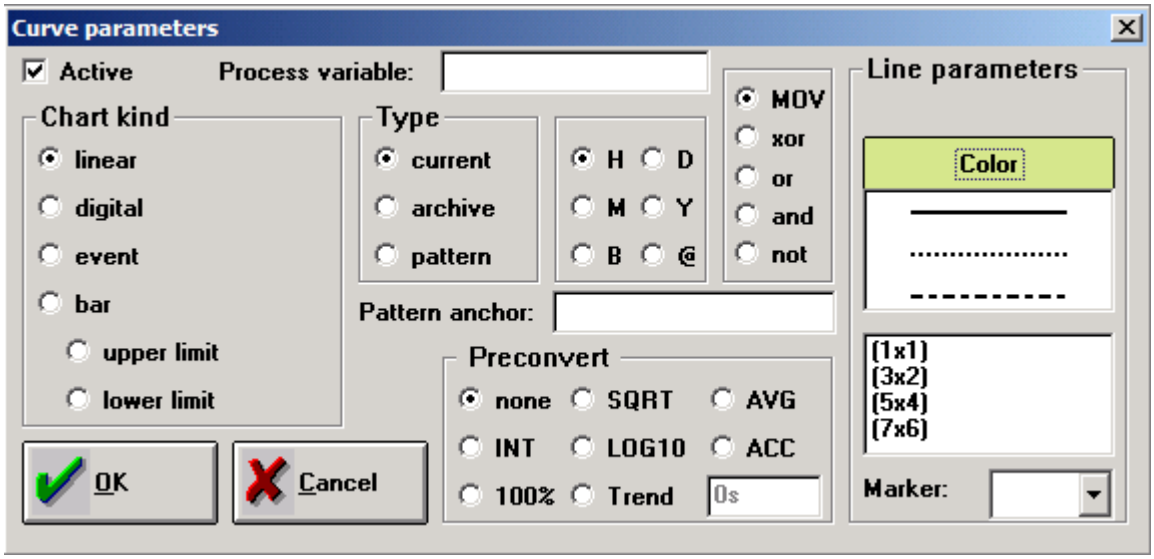


Figure 18-87. The 'Curve Parameters' Window.

Active - an option to determine if the curve is active on starting of the object.

Chart kind - a group of boxes to select the chart type:
linear: permits to present the value runtime as a broken line, the nodes of which are the measurement points connected with a straight line.
digital: permits to present bistate signals, e.g. switching ON/OFF actions of actuators.
event: permits to show moments of certain events determined as the values from the range from 0 to 10 by displaying graphic

markers on the argument axis (non implemented).
bar: permits to present variations of a value in the form of a sequence of bars, the height of which corresponds to the value. Two additional chart kinds are used in conjunction with it:

- *upper limit****: determines the method of presentation of exceeding of the upper limit;***
- *lower limit****: determines the method of presentation of exceeding of the lower limit.***

Process variable **-** to specify a name of the variable to be presented by means of the given curve.

Type **-** a group of boxes to determine a curve type. A current curve causes shifting of the curve region contents in function of the time elapsed;
Default = current
A group of boxes to determine an archive of the variable values. Available there are archives of type H, M, D, Y and B (see description of the ASPAD unit);
Default = H
A group of boxes to determine a logic function which is used to draw a curve on the screen. It serves to obtain special effects on the charts;
Default = MOV

Pattern anchor **-** field allowing indicating the starting point for the pattern curve on the OX axis. Concerns only the pattern curves.
- if the field is empty, shifting the pattern curve is performed with help of archival curves handling window
- field may contain name of **variable,** in which the time moment defining the point of pattern curve anchoring is stored. The variable must be of **DW** type**.** Operations on the pattern curve performed by functions of archival curves handling cause automatic inserting the point of pattern curve anchoring into discussed variable. After new opening the diagram with the chart or application restart, last point of pattern curve anchoring is saved in the variable.

Preconvert **-** a group of boxes to determine a type of preconvert of the variable value before drawing. The following preconvert modes are available:

none	without preliminary conversion.
INT	integer.
LOG10	decimal logarithm.
SQRT	square root.
AVG	a „walking average" of the value for the preset time horizon; the zero value means no measurement.
ACC	a „walking average" of the value for the preset time horizon; the

	average value ignores the no measurement situation.
Trend	gradient calculated in the set time horizon.
100%	percentage within the variable variation.
BIT	- allows to draw a bi-status chart from any process variable bit; the function parameter is the number of bit, which is checked and decides about the function value; BIT function returns 0, if the bit is not set, and 1 in the opposite case;
BITV	- allows to draw a bi-status chart from any process variable bit; the function parameter is the number of bit, which is checked and decides about the function value; BITV function returns 0, if the bit is not set, and a value equal to bit number in the opposite case - this allows to display several bi-status curves on one chart (even for the same variable).

Default = **none**

Color **- Line color**; Default = green
Line type; Default = solid
Line thickness; Default = 1x1

Marker **-** a type of the marker drawn at the measurement point.

18.4.38.7 Setting Parameters from Variable Definitions Database

Receiving the Values of Parameters from VarDef
Determining an Archive of Variable Values
Defining the Legend Format
Preconvert of the Variable Value in 100% Mode

GRAPH object parameters (from the 3.05.006 version of the AS program) can be partly set on the base of values form VarDef.

The information stored in the VarDef are used to:

- creation of a chart title;
- 0Y axis scaling;
- creation of the legend of a chart;
- determining an archive from which the variable values are received;
- preconvert of the variable values in 100% display mode.

Receiving the Values of Parameters from Variable Definitions Database

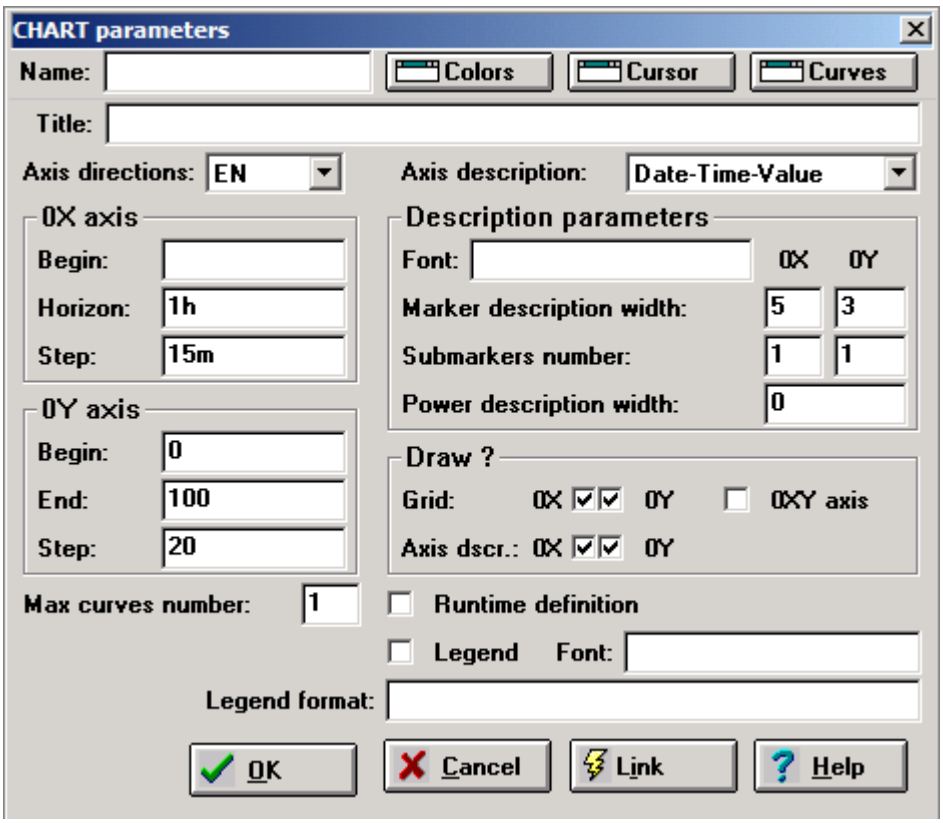


Figure 18-88. The 'CHART Parameters' Window.

The values of the fields: in *0Y axis* group, *Marker description width*, *Submakers number* and *Title* are set from the VarDef.

The *Link* button inserts data (with the @ marker) into the mentioned fields from the VarDef unless the value is set manually.

The principles of determining the value of parameters are following:

Begin - is loaded from the *DisplayRangeFrom* (*Display range from*) field; minimum from the values of all curves;

End - is loaded from the *DisplayRangeTo* (*Display range to*) field; maximum from the values of all curves;

Step - is loaded from the *DisplayRangeStep* (*Display range step*) field; maximum from the values of all curves; if the parameter isn't found in the VarDef, it is calculated in accordance to the occurring algorithm: *Begin – End* / divided in succession through 3, 4 and 5. The range of variability is divided by divider which gives the rest *0* (as first) – and in this way a step is determined. If no divider is fulfilling the condition, the divider equal 2 is applied.

Maker description width - is loaded from the *DisplayRangeWidth* (*Display range width*) field; maximum from the values of all curves; if the parameter isn't found in the VarDef, it is calculated in accordance with the occurring algorithm:
Width = a number of characters in *Begin* or *End* field (the higher value is chosen) +1;

0Y submarkers number - is loaded from the *DisplayRangeDivision* (*Display range division*) field; maximum from the values of all curves; if the parameter isn't found in the VarDef, it is calculated in accordance with the occurring algorithm: *Step* / divided in succession through 3 and 4. The first divider which gives the rest *0* determine the number of submarkers. If no divider is fulfilling the condition, the divider equal 2 is applied.

Title - is loaded from the *WykresTytul* (*Chart title*) field of the variable being displayed as first curve; the @ marker is required to be set manually in the *Title* edition field.

Determinig an Archive of Variable Values

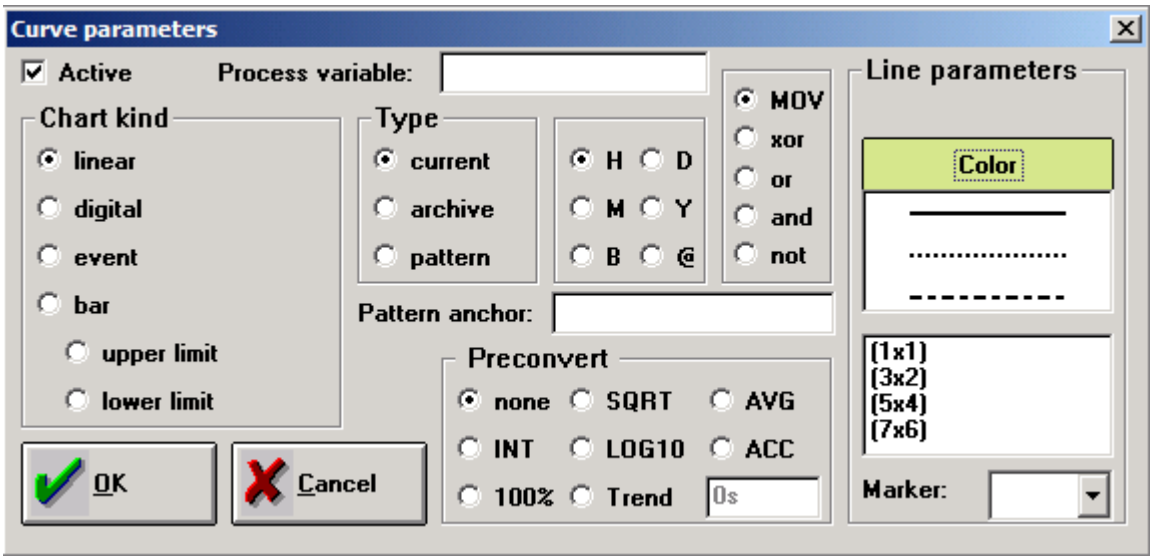


Figure 18-89. The 'Curve Parameters' Window.

The type of @ archiving means, that real type of the archive using when opening the access to the variable, will be the same as defined in the *ArchivingParameters* (*Archiving parameters*) attribute in the VarDef.

Defining the Legend Format

Format of the legend displayed below the curve region is freely defined. It is necessary to declare the text defining the legend in *Legend format* edition field of the parameters window of CHART object. The parameters being loaded from the VarDef are declared through names of parameters written in braces, e.g.:

{*Name*} -- {*Unit*}

If *Legend format* field remains empty or the base of variables isn't used, the standard format will be applied:

Name Range Description.

It is also possible to put the stamp {-} in *Legend format* to display a dash of the curve colour in the legend.

Convertion of the Variable Value in 100% Mode

For the curves with applied option of *'preconvert of the variable value in 100% mode'* the variation range is loaded from *MeasurementRangeFrom* (*Measurement range from*) and *MeasurementRangeTo* (*Measurement range to*) fields from VarDef. If the database of variables isn't available the range of the variable is determined on the basis of parameters of ASMEN's conversion functions (*ConversionFunctionRangeFrom* – field: *Conversion function range from* and *ConversionFunctionRangeTo* – field: *Conversion function range to*).

19. Options Defined in asix.ini File

The **[STATE]** section is **used in *asix.ini* file**. Entries in this section concern the operation parameters of the AS32 program and are used also for the recent program settings saving.

The asix.ini file is located in the main **asix** package instalation directory.

Example:

```
[STATE]
DESIGNER_WINDOW=83,488,103,270
NUMBER_OF_COPIES=1
```

FONTS_EXAMPLE = text

Meaning - the definition of the text used in the window of font selection to display an example of use of a chosen font.
Defining - by means of the font selection window.

NUMBER_OF_COPIES = number

Meaning - item setting a number of created backup copies. It concerns initialization files of applications and files describing the masks.
Default value - one backup copy is created.
Defining - manually

EDITOR_WINDOW = left,right,up,down

Meaning - the item saving the last location of the Editor window. Coordinates expressed in pixels, the point 0,0 lies in the left-upper corner of a window.
Default value - standard position
Defining - automatic

DESIGNER_WINDOW = left,right,up,down

Meaning - the item saving the last position of the Designer window. Coordinates expressed in pixels, the point 0,0 lies in the left-upper corner of a window.
Default value - standard position
Defining - automatic

VIEWER_WINDOW = left,right,up,down

Meaning - the item saving the last position of the window for Reports display. Coordinates expressed in pixels, the point 0,0 lies in the left-upper corner of a window.
Default value - standard position

Defining - automatic

REPORT_WINDOW = left,right,up,down

Meaning - the item saving the last position of the report system window. Coordinates expressed in pixels, the point 0,0 lies in the left-upper corner of a window.
Default value - standard position
Defining - automatic

LAST_FILE = file_name

Meaning - the item saving the last used initialization file of an application. At the moment of the AS32 program start in the designer mode, the initialization file with a given name will be loaded.
Default value - *noname.ini* file is to be loaded
Defining - automatic

VIEW_POSITION = left,up,right,down

Meaning - the item to save the last location of mouse position window. Coordinates are expressed in pixels; the point 0,0 lies in the left-upper corner of the screen.
Default value - the position window is not used
Defining - automatic

SEQUENCE_DEMO = [YES|NO|AUTO]

Meaning - the item permits unlocking of demo sequence mechanism. Additionally, the use of *AUTO* symbol causes an automatic start of sequence replay at the moment while the application starts.
Default value - demo sequences are not used.
Defining - manual.

CURSOR_SNAP = stepX,stepY,pointX,pointY

Meaning - the item provided to save the last used parameters of a cursor snap.
Default value - step equal to 1 towards both directions
Defining - automatic, at the moment of a step change

20. Architect - interactive environment for application configuration

Introduction of Architect module is a true breakthrough in application development. It enables a fully visual designing, configuration and ongoing editing of **asix** system application with use of dialogue boxes with system of tabs. The tabs group all options responsible for specific functional areas of the application in an orderly and clear manner and unfailingly lead the designer through the process of correct declaration of the required parameters. Architect module, associated with VarDef module responsible for storage of variable definitions databases enables a fully interactive and visual handling of the definitions base, including: creation of base structure and editing its contents.

VARDEF – VARIABLE DEFINITIONS DATABASE

Asix5 package is fitted with a new variable definitions database module, VarDef, which is fully supported by Architect.

VarDef manages the databases of Microsoft SQL Server 2000/2005 and of MDB format (Jet / Microsoft Access). To enable migration of application from earlier **asix** versions, the variable definitions data may be converted from Paradox database or text files to MS SQL or Jet base format.

Architect module enables a fully interactive and visual handling of VarDef databases in Jet or MS SQL format. In this scope, it combines the functionality of two separate programs used previously for creating and editing the variable definitions database: Variable Database Manager and Variable Database Editor (used in older versions of asix package).

The configuration data pertaining to the variable definitions database are configured in Architect with use of *Items definitions editor* (opened with *Databases* button located in the left block of Architect window) and saved in the XML configuration file of the created application.

See more information in:
Architect user's manual (Architect.pdf located in the main **asix** package instalation directory: C:\asix\manuals\Architect.pdf)

21. AsAlert - system for remote notification of alarm events

AsAlert module is used for remote notification of selected persons about important events by means of e-mail or SMS messages.

The messages can reach the addressees by means of different transmission methods:

- as standard e-mails through the Internet and SMTP protocol;
- as standard e-mails through a GSM mobile network with use of mail services provided by mobile carriers;
- as SMS messages for mobile phones through a GSM network;
- as SMS messages for mobile phones through the Internet.

AsAlert supports three types of recipient's addressing:

- standard user, addressed by mobile phone number or e-mail address;
- user groups – allowing the sending of an alert to multiple addressees at one time;
- schedules – mechanism enabling selection of addressee depending on the time of alert sending.

Client applications demanding alert sending may operate on network stations. Connection with AsAlert is realized after favourable verification of access authorizations – verification is made by AsAlert on the basis of own list of users and their passwords.

Detailed information on using the AsAlert program may be found in - Alerter.hlp (AsAlert.pdf), placed in the main directory of installed package.

22. AsAudit - Solution for Systems Subject to Strict Validation Procedures

Asix5 package is also appropriate for applications, which need to meet specific validation requirements, in accordance with **GAMP4, FDA 21 CFR Part 11** regulations applied in pharmaceutical and food processing industry. The module extends the central user log-in and authorization control of system with registration of performed controls, operator's action and application integrity.

AsAudit module supports the following functions:

- User log-in and authorization control system
- Operator's notepad
- Logging control actions performed for selected variables
- Logging the operator's actions
- Application integrity control

AsAudit operation is based on application of SQL database for storing the configuration data and logging data collected during application operation.

See: *AsAudit – User's Manual*

(AsAudit.hlp in the main **asix** package instalation directory:

C:\asix\helpen

or

AsAudit.pdf located in:

C:\asix\manuals\AsAudit.pdf)

23. Multilingual Applications

The multilanguage applications system allows the application operation language to be switched over at the operator's request. Every text entered by the designer on the application development stage is subject to switching over. Set of application languages in use is declared in the application parameters. Every language to be used must by based on the 256-character set.

Typically, the application is written in the basic language. On a further stage of works, translations of texts used in the application are delivered.

You should not confuse the languages of application operation with the language of program operation. The program operation language is used in dialog windows and messages. The program operation language can be Polish or English. The program operation language is defined on the basis of system settings or selected by means of SelectLanguage program. The program operation language is the same over the whole time of operation.

23.1. Declaration of Application Languages

The language switchover system is active when at least two operation languages are defined in the application configuration file. Languages are defined in ***Languages*** parameter with use of Architect module:

Architect > *Fields and Computers* > *Start parameters* module > *Languages* tab

For a given application five languages may be declared, which means that the application built using **asix** package can work in five languages. The first language is of special significance. It defines the so-called primary language. It is the language the application was built in.

23.2. Selection of Application Language

The application language can be changed by execution of LANGUAGE function available from OPTIONS menu in Control Panel window. This function opens the box with a list of defined application operation languages. Contents of the list are created on the basis of ***Language*** parameter (see: *23.1. Declaration of Application Languages*).

An alternate language selection mechanism is the LANGUAGE operator action with the following syntax:

LANGUAGE *[language_code]*

Execution of the action with the code given results in switching the operation language over appropriately. Lack of the parameter results in opening the language selection box.

23.3. Translation Mechanisms

The following mechanisms are used to translate texts:
- Multilanguage Texts;
- Text Table;
- Translation Table.

23.3.1. Multilanguage Texts

The general mechanism to allow definition of different application language versions is based on appropriately formatted text.

The format of such multilanguage text is as follows:

[*code_1*]*text_1*[*code_2*]*text_2*[*code_n*]*text_n*

The language codes in [] brackets have to be consistent with codes of the languages defined in [LANGUAGES] section. During the program operation, variant suitable for the current application language is selected from the multilanguage text. If variant for the current application language is missing, then variant of the text declared as the first is used.

Multilanguage texts can be used as an independent translating mechanism or as a component of another mechanism.

NOTE If character [needs to be used in the text, it should be entered as a sequence of \[.

23.3.2. Text Table

The Text Table mechanism is used for translating the texts, which are displayed as a part of application components and are not given openly by the designer (they are built into the program code).

This mechanism is run when TextTable.Lng file is placed in the start-up directory.

The Text Table file is a text-type file and its individual lines are multilanguage texts that determine translations of the texts. Commentary lines are acceptable in the file. These lines begin with semicolon.

The **asix** package comes with TextTable.Tlt file containing a complete table of texts for Polish and English languages. The designer should copy this file into the application directory named TextTable.Lng, and then supplement the multilanguage texts with variant definitions for the other application languages used.

The rule of the Text Table operation is to search translation for a text built into the program from among the multilanguage texts defined.

23.3.3. Translation Table

Generating the Translation Table File

The Translation Table mechanism is used for translating the texts entered by the operator during definition of the application. It mainly concerns the texts from objects placed on visualisation masks.

This mechanism is run when TranslateTable.Lng file is placed in the start-up directory. It is a text-type file and its individual lines are multilanguage texts. The texts are divided into groups, which split the texts by the place of their application. The beginning of the group is marked by the following line:

```
#group_name
```

The following group names are accepted:
- OBJECT – texts from visualisation objects,
- MASK – texts used in visualisation mask definitions,
- TIMER – texts from timer definitions,
- TREND – texts from definitions of trends built into AS program,
- TABLE – texts from table definitions.

Commentary lines are also acceptable in the translation files. These lines begin with semicolon.

The rule of the Translation Table operation is to search translation for a text used in the application definition from among the multilanguage texts defined. The text is searched in variant for the primary application language and the variant is selected for the current application language.

Generating the Translation Table File

The base Translation Table file can be generated automatically using the TRANSLATION_FILE function available from TOOL menu in the Designer window. Execution of this function results in displaying the following dialog window:

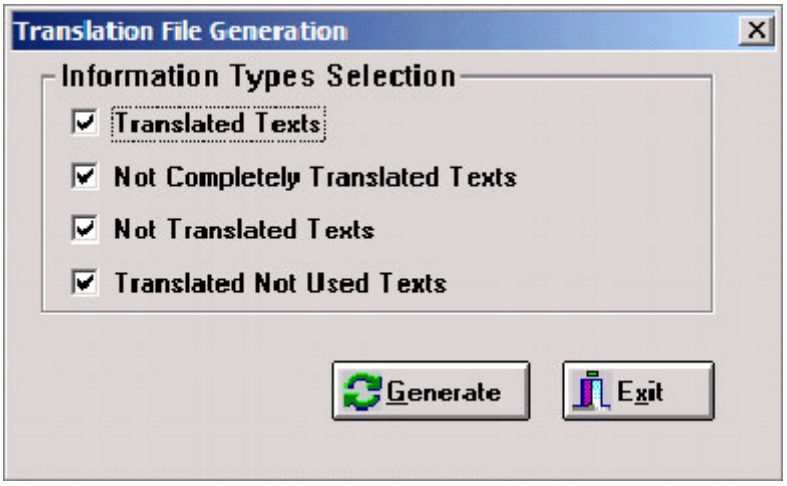


Figure 23-1. The Translation File Generation Window.

After the category of generated information is selected and *Generate* button is pressed, all the texts found will be saved to TranslateTable.Tlt file. If TranslateTable.Lng file has existed before, then all translations contained in it will be transferred into the generated file.

The designer should add the missing translations directly in TranslateTable.Lng file or supplement TranslateTable.Tlt file and rename it TranslateTable.Lng.

23.3.4. Variables Database Substitute Attributes

The variables database substitute attributes mechanism is used wherever the displayed texts are retrieved from the attributes of process variables. In addition, this mechanism is used in dialog windows used to select the variable and the attribute. The mechanism ensures that the attribute language version suitable for the current application language is retrieved automatically.

The substitute attribute is an attribute that replaces another one when the application language is changed. First the attribute that acts as a substitute should be defined in application configuration file with use of Architect program:

Architect > *Databases* > *Varbase* > *Scheme editor* (variable definition database should be opened) > *Substitute attributes* tab

See: Architect – User's Manual, chapter *2.2.2.5. Substitute attributes*.

Attributes and substitute attributes should have declared the language.

Architect > *Databases* > *Varbase* > *Scheme editor* (variable definition database should be opened) > *Attributes languages* tab

See: Architect – User's Manual, chapter *2.2.2.6. Declaring the attribute languages*.

Restrictions on defining substitute attributes

1. Substitute attributes cannot be defined for basic attributes, such as *Name, Channel, Address, NoOfElements, SamplingPeriod, ConvertingFunction, Archive, ArchivingParameters, Group*, and for *AsmenaFile, AspadaFile, DataSource, DataTable* attributes.
2. Substitute attributes cannot be parameters of the converting function.
3. Substitute attribute must be of the same type as the substituted attribute.

23.4. Change in Coding Pages of Fonts

If the names of coding pages have been defined in declaration of application languages, the automatic change in coding pages for certain fonts will be carried out. Only the fonts defined in the initiate file and description of which includes the coding page consistent with the coding page of the primary language are subject to change.

The following names of coding pages are accepted: ANSI (West-European), BALTIC, EASTEUROPE, GREEK, RUSSIAN, TURKISH, CHARSET_*number*.

23.5. Handling of Application Components

This chapter provides detailed information on handling of multi-languages for individual components of application definition.

23.5.1. Application Parameters Declared With Use of Multilanguage Texts

Selected items in the initiate file can be defined using multilanguage texts.

These are the following items related to printing:

Application name
Header
Report header
Footer
Report footer
Historical alarms header
Active alarms header
Historical alarms footer
Active alarms footer

The above parameters are set up with use of Architect module:

Architect program > Fields and Computers > Printout module

It is also possible to use multilingual texts declared as ***Text parameters***:

Architect program > *Fields and Computers* > *Masks* module > *Text parameters* tab

Whether a multilanguage text is handled in a specific item depends on the context of the use of this item.

23.5.2. Operator Actions

ASK action *(see: 16.2. Description of Actions)*
The ASK action parameter can be a multilanguage text.

MENU action *(see: 16.2. Description of Actions)*
The texts displayed in menu, defined in pum file, may be given in multilanguage text format.

23.5.3. Scripts

Creation of the script that considers the current application language is the exclusive responsibility of the script initiator. The current language code can be retrieved from LanguageApplication properties in Application object

23.5.4. Reports

When necessary, you should prepare various language versions and select the appropriate report in Report window.

23.5.5. Built-in Trends

Parameters *Description* and *Title* of trend definition are translated by means of Translation Table. Constant texts in trend window are translated by means of Text Table.

23.5.6. Visualisation Masks

Parameter *Mask Description* is translated by means of Translation Table.

23.5.7. Visualisation Objects

For objects retrieving text parameters from the variables database the mechanism of substitute attributes may be used. The remaining rules of translation are as follows:

- DATE+TIME

The names of months are retrieved from Text Table

- MESSAGES

The message texts are defined in the text-type file where multilanguage text format can be used. If a message is not a multilanguage text, it is translated by means of Translation Table.

- DRIVE

Constant key descriptions are retrieved from Text Table Descriptions given by the designer are translated by means of Translation Table.

- SWITCH

Texts from the *List of Texts* are translated by means of Translation Table.

- BUTTON

Parameters *Line 1* and *Line 2* are translated by means of Translation Table.

- SELECTOR

Parameter *Text* is translated by means of Translation Table.
Texts from the *List of Texts*, unless text equivalents are used, are translated by means of Translation Table. In case of text equivalents the multilanguage texts in the initiate file can be used.

- TEXT

Parameter *Text*, unless it is text equivalent, is translated by means of Translation Table. Every text line is translated individually. In case of text equivalents the multilanguage texts in the initiate file can be used.

- TEXTS

Texts from the *List of Texts*, unless text equivalents are used, are translated by means of Translation Table. Every line in a multi-line text is translated individually. In case of text equivalents the multilanguage texts in the initiate file can be used.

- GRAPH

Parameter *Title* is translated by means of Translation Table.

- OFF SWITCH

State descriptions are translated by means of Translation Table.

23.5.8. Tables

Parameter *Title* of table definition is translated by means of Translation Table. For parameters retrieved from the variables database substitute parameters may be used.

23.5.9. Alarm System

Texts of alarm messages and group names in text-type definition files may be given in multilanguage text format.

Built-in texts, used when printing the alarms, are translated by means of Text Table.

24. AsBase

There are some classes of applications, when archiving mechanisms described in the *ASPAD – Data Archiving Module* chapter are insufficient; these are mostly the applications that use concept of recipes or applications related to monitoring of material flow, with the necessity of asynchronous logging of the entire sets of process variables at predefined events.

Execution of applications with such requirements in **asix** environment is made possible in a flexible and versatile manner through the use of scripting; the method requires users to show the knowledge of ADO objects handling, SQL query formulation and use of **asix** variables on the level of scripts**.** These demands are eliminated by **AsBase** module, which disguises all nuances of database access and provides a set of predefined operations and dialogue boxes, thus creating an environment for use of databases in a manner focused on definition and selection of recipes or archiving and review of the sets of parameters related to the occurrence of defined events.

The scope of the **AsBase** module functions includes:

- embedded interactive application-making system,
- access authorization integrated within an application,
- automatic archiving of process data launched upon time schedule or on the basis of process variable values,
- option of manual archive editing (supplementing),
- recipes management, including definition of recipe values, loading a recipe value into the process variable and logging of recipe loading operations,
- view of tag current values, with the option of manual launch of recording to the application database,
- system for analysis and printing of archived data and recipes,
- data export to text files or XLS, XML and HTML files, built-in database management system.

Detailed information on using the AsBase program may be found in - AsBase.hlp (AsBase.pdf), placed in the main directory of installed package.

25. AsComm Link Manager

AsComm Link Manager program is designed for management and monitoring links created by modules of **asix** system. The links are created with use of serial ports or modems (switched lines). The basic objects managed by AsComm are "Client" and "Resource". Client is a module of **asix** system that uses AsComm functions. Client is identified by its name. Resource is a communication link such as serial port or modem. Using the AsComm program, "client" may use "resources" to create links. One of the functions of AsComm program is sharing the communication link between several clients. The way of resource sharing and performing the other functions is defined by appropriate configuration of the initialization file.

Detailed description of application of AsComm Link Manager may be found in AsComm.hlp (AsComm.pdf).

26. Asix4Internet - Visualization and Control via Internet

asix includes tools enabling creation of Internet applications with a view of technological process dynamically refreshed in the browser window. The starting point for creation of dynamic applications is the set of so-called *web-services* supported in the **asix** environment (in AsixConnect Server package), which provide a full array of process data on the Internet application server – in particular the current data, archive data and alarms.

Asix4Internet provides access to process information on Internet with the following modules:

- *AsPortal* - Portal of Process Information, ready for immediate use when connected to **asix** application. AsPortal displays application process database, current process values, current and historical alarms as well as data trends either in tables or grahic charts, presented in Internet browser window;
- *As2HTML* - library of scripts and CSS style sheets which supports design of process visualization for Internet browser window;
- *As2WWW* - set of tools for automatic conversion of **asix** application into IE 6 browser visualization.

Detailed information on using the components of Asix4Internet may be found in - Asix4Internet.hlp (Asix4Internet.pdf), placed in the main directory of installed **asix** package.

27. AsixConnect 5 Package

AsixConnect 5 Package is a package of servers extending applications of **asix** package such as monitoring and computer supervision of industrial processes. AsixConnect includes OPC, Automation and DDE servers that allow access to the current values of process variables from the database of applications of **asix** program and Automation server that allows access to the archive values of process variables. AsixConnect is an integral element of **asix** package but is also supplied as separate product for applications in PC stations connected to local area networks and having access to the data servers provided with **asix** packages. In this case AsixConnect allows, in Windows environment, access to data imported from remote computer stations provided with links to process controllers.

Detailed information on using the AsixConnect program may be found in AsixConnect5.hlp (AsixConnect5.pdf).

28. AsTrend Program

AsTrend Program is designed for graphical presentation of archive data collected by ASPAD data archiving program. The two modes of operation of this program are possible: static one, for presentation of data collected in archive files or dynamic one, for displaying data retrieved on-line from process (simulation of recorder). In the second mode, operation of AsTrend is similar to that of CHART object but using AsTrend you can change the list of trends to be displayed, parameters of value and time axis of chart, make printed copy of trends and make on trends many other functions that are not possible with CHART object. In comparison to Trend tool accessible from designer menu, AsTrend is easier to use and configure and is provided with more features for data presentation. In new applications AsTrend program is then recommended.

Detailed description of application of AsTrend program may be found in AsTrend.hlp (AsTrend.pdf).

29. AslView Program

AslView Program is designed for monitoring and diagnostics of network links in **asix** system. AslView program allows to create connection to ASLINK network module running on the local or remote station and to monitor the status of links that other objects of **asix** system, such as ASMEN and ASPAD have created. It enables also monitoring the status of time synchronization with other stations. With AsIView program you can configure operation of ASLINK network module.

Detailed description of application of AslView program may be found in AslView.hlp (AslView.pdf).

30. AspadTools Program

AspadTools Program is designed for treatment and maintenance of archive files of Y, M, D and H type.

It enables:

- changing the conventions of file names from an old one (ASPAD 5) to a new one (ASPAD 6) and vice versa,
- conversion of archives to other type (e.g. M to D),
- exporting to text file,
- testing and repairing the archive,
- collecting the statistics on quantity of data for the individual archived variables,
- changing the name or deleting the variable from archive file.

Detailed description of application of AspadTools program may be found in AspadTools.hlp (AspadTools.pdf).

31. Manager of Logical Channels for the OPC Driver

Manager of logical channels enables easy editing the definition of transmission channels using the OPC server through simple setting options in the dialog window specially prepared for the OPC driver. The text edition of transmission channels using other drivers than the OPC driver is also possible.

The manager is placed in ChannelsManager.exe file in the directory of the **asix** system (by default c:\asix). When running the program the occurring window is opened:

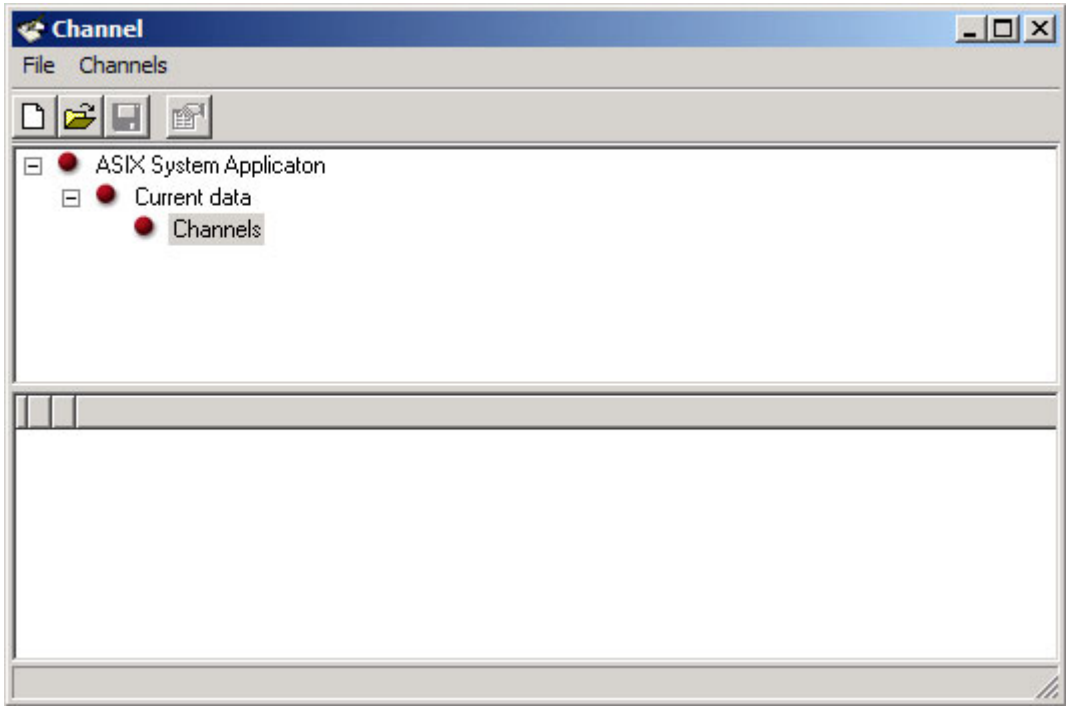


Figure 31-1. The Channel Manager Main Window.

There are the following commands on the ***File*** menu:

New	- creates a new ini file of the **asix** application;
Open	- loads existing ini file into the manager;
Save	- saves all modifications in ini file;
Save as	- saves ini file as a new file;
Exit	- ending operation of the ChannelsManager program.

There are the following commands on the ***Channels*** menu:

New	- creates a new logical transmission channel;
Rename	- renames the transmission channel;
Edit	- edition of the channel definition;
Edit as text	- edition of the channel definition in the text format (also edition of the channel referring to the OPC server);
Delete	- removing the channel definition.

When opening the ini file the following window appears:

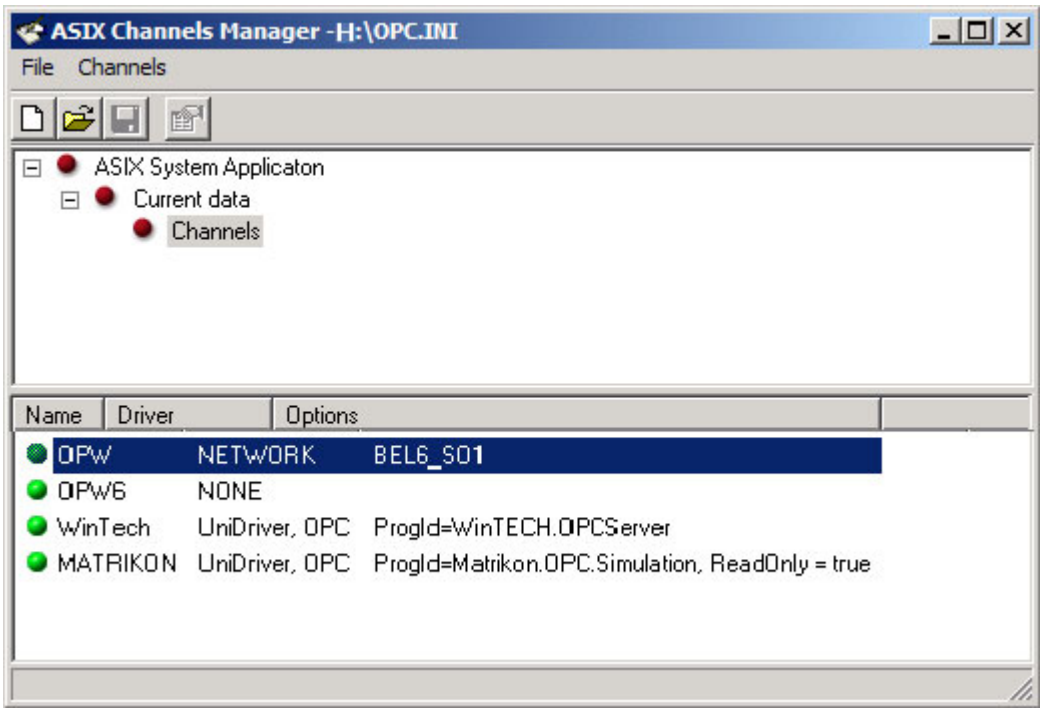


Figure 31-2. The Channel Manager Window.

WinTech and MATRIKON channels use the OPC driver. The window used to edition of the WinTech channel is presented in the following figure

Figure 31-3. The Channel Options Window.

The OPC driver identifier may be inserted into the *OPC server identifier* field manually or chosen from the extended menu. The register of the Windows system is looked through in order finding registered OPC servers before first

displaying the *Channel options* window. If such servers are found their identifiers appear on the extended menu of the *OPC server identifier* field.

Remaining options that are available in the dialog window are characterized in the chapter *ASMEN – Drivers and transmission protocols/OPC driver*.

There are two transmission channels using NETWORK and NONE drivers in the exemplary application. The window editing the OPC channel looks as in the following figure.

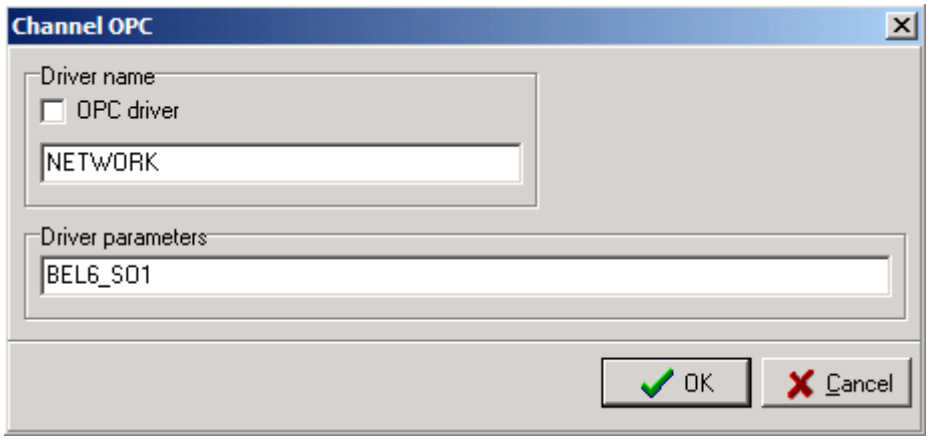


Figure 31-4. The 'Channel OPC' Window.

32. Dispatcher Memo

The Dispatcher Memo Module was developed with focus on dispatcher's systems. Dispatcher Memo is a program that facilitates recording of operation notes for dispatchers. First of all it replaces the common paper notebook providing such features as possibility of corrections, automatic numbering, adding time stamp and author's name.

Features of Dispatcher Memo that are not available in common notebook, are as follow:

- fast searching the notes;
- searching the notes according to given criteria;
- creating reports from data base of notes.

Detailed information on using the Dispatcher Memo program may be found in Notatnik.hlp (DispatcherNotebook.pdf).

33. As_Power_Close

As_power_close program is designed for cooperation with software supervising an operation of UPS power supply units. When As_power_close program is started, it causes immediate and unconditional breakdown of AS program. In AS program message log file the information on reason of breakdown is written.

As_power_close.exe file is included in **asix** installation directory.

34. ZMIANADP Program

Process Variables Names Conversion Program ZMIANADP can be used to globally exchange variables names used as parameters of visualization objects. The program should be used for two purposes:

- complete names exchange, e.g. when similar application is built,
- names exchange for single mask, e.g. creation of new mask based on another similar mask.

ZMIANADP program is executed in batch mode. It is started by the following command:

zmianadp <names_file> <mask_name>

where:

<*names_file*> - parameter declares text file name, in which variables names conversion information is written;

<*mask_name*> - parameter declares mask definition file name, which contents should be changed. Wildcard characters ? and * can be used in <mask_name>. This allows to convert many files in single operation.

<names_file> - is a text file in which every line consists of two process variables names separated by space character. First name defines variable's name that should be changed. Second name in the line defines new name which should be used in place of the first one.

EXAMPLE

Changes.txt file contents:

TEMPA11 TEMPX30
TEMPA12 TEMPY0

Command:

zmianadp changes.txt *

Result:

For all masks in current directory following conversions will be done:

- every occurrence of *TEMPA11* variable will be changed to *TEMPX30*
- every occurrence of *TEMPA12* variable will be changed to *TEMPY0*

35. Pattern Trends

asix system offers a wide range of possibilities for pattern trends:

- **PEdit** program for pattern trends edition;
- possibility for automatic changing a point of locating the pattern trend beginning on the chart (pattern anchor);
- possibility for pattern anchor storage, also after **asix** restart.

ASKOM offers also solutions based on scripts, supplemented with **PSelect** program. They allow:

- manual selection of pattern trend curve;
- automatic selection of pattern trend curve;
- transfer of chosen pattern curve to the controller.

They are treated as an additional, read-only archive type, marked with letter P.

Pattern data are stored in a database analogous to B-type archive. The best solution is to place the patterns in a separate database treated as a separate ASPAD program resource. It is possible to store the B and P archives in the same database, but it's not recommended.

Detailed information on Pattern Trends may be found in - PEdit.hlp (PatternTrends.pdf), placed in the main directory of installed **asix** package.

36. SCRIPT Module

The script module is designed for extending a range of functions of **asix** system to:

- non-standard calculations that may be performed on process variables,
- programming the non-standard reaction on internal events of **asix** system,
- exporting the values of variables and the other information on **asix** system,
- retrieving the data from non-standard data sources.

To perform above functions the script module enables:

- automatic running a number of concurrent scripts developed in different programming languages with the possibility to define a priority of script,
- data exchange between scripts,
- calling from a given script functions contained in other scripts developed in different programming languages (library scripts),
- passing control to **asix** system as a result of events arising during script execution (operator's actions),
- controlling the script execution time,
- tracking the changes in the file including script program and its automatic execution after detecting such changes.

Detailed description of application of SCRIPT module may be found Skrypty.hlp (Scripts.pdf).

37. Simulator

Data Simulator Module enables creating realistic simulations of industrial processes. It is used in demo systems, that operates without access to process controllers. The main difference to internal ASMEN's simulations is creating logical elements of a number of mutually interconnected variables. Data Simulator Module is activated automatically when the [SIMULATION] section in application configuration file is detected.

The simulation description consists of set of the element descriptions (in successive items of the section) that define a way of variable changing and their mutual links.

The [SIMULATION] section is declared with use of Architect module in the following way:

- open Architect program > *Fields and Computers* > *Miscellaneous* module > *Directly entered options* tab (the tab is available when the *Show Advanced Options* from Architect window > *Fields and Computers* menu is switched on);
- enter *Section name*, *Option name* and *Option value* for the following options:

Section name	- SIMULATION
Option name	- **INIT**
Option value	- *variable,value*
Meaning	- variable initialized to the chosen value
Parameters:	
variable	- name of a process variable;
value	- value to be entered to a variable.

Section name	- SIMULATION
Option name	- **MOTOR**
Option value	- *control,status,setting,value,range*
Meaning	- drive simulation according to an action of coupled objects of MOTOR and SLIDER types.
Parameters:	
control	- name of a control variable of the MOTOR object;
status	- name of a monitored variable of the MOTOR object;
setting	- name of a control variable of the SLIDER object;
value	- name of a monitored variable of the SLIDER object;
range	- the upper limit of the monitored variable of the SLIDER object.

The object calculates a value of the monitored variable of the MOTOR object depending on changes of its control variable. Simultaneously it causes a respective change of the monitored value of the SLIDER object. The achieving the limit value influences reciprocally the monitored variable status of the SLIDER object. It is also possible to change the value of the SLIDER object through its control variable setting (displacement of a slider on a diagram).

Section name	- SIMULATION
Option name	- **RANDOM**
Option value	- *variable,lower_limit,upper_limit,frequency*

Meaning - pseudo-random value generation in a given range.
Parameters:
variable - name of simulated variable;
lower_limit - lower limit of change range;
upper_limit - upper limit of change range;
frequency - interval in seconds between changes;

Section name - SIMULATION
Option name - **REGULATOR**
Option value - *current_value,set_value,deviation_variable,reach_time*
Meaning - controller simulation
Parameters:
current_value - name of a controlled variable;
setting_value - name of a variable containing a set-point value;
deviation_variable - name of a variable of a deviation value;
reach_time - time in seconds, after which the current value achieves the set-point.

A set-point value change causes suitable modification of a variable current value with simultaneous calculation of the deviation of regulation.

Section name - SIMULATION
Option name - **RELATION**
Option value - *variable,lower_limit,upper_limit,control_variable,range*
Meaning - proportional dependence
Parameters:
variable - name of a dependent variable;
lower_limit - lower limit of the variation range of a dependent variable;
upper_limit - upper limit of the variation range of a dependent variable;
control_variable. - name of a control variable;
range - maximum admissible value of a control variable.

A value variation of the control variable modifies a dependent variable. A change of the dependent variable is linearly proportional to a value of the control variable, taking into account variation of the control variable within the range from 0 to *range*, and the counted variable within the range from *lower_limit* to *upper_limit*.
variable = *lower_limit* + (*upper_limit* - *lower_limit*) * (*control_variable / range*)

Section name - SIMULATION
Option name - **OR**
Option value -*Y,YB,X1,XB1,X2,XB2[,X3,XB3]*
Meaning - logical OR of the chosen bits of two or three variables.
Parameters:
Y - result variable - a 16-bit word;
YB - bit mask**,** determining set/zeroed bits of the result variable given as a decimal integer number;
X1,X2,X3 - variables for which the logical OR is performed, 16-bit words;
B1,XB2,XB3 - bit masks that obtain bits taken into account, decimal integer numbers. Parameters *X3* and *XB3* can be omitted

Section name - SIMULATION
Option name - **AND**
Option value - *Y,YB,X1,XB1,X2,XB2[,X3,XB3]*
Meaning - logical AND of the chosen bits of two or three variables.
Parameters:
Y - result variable - a 16-bit word;
YB - bit mask**,** determining set/zeroed bits of the result variable given as a decimal integer number;
X1,X2,X3 - variables for which the logical AND is performed, 16-bit words;
XB1,XB2,XB3 - bit masks that obtain bits taken into account, decimal integer numbers.

Section name - SIMULATION
Option name - **COMPARISON**
Option value - *Y,X,L*
Meaning - comparison of a variable value in relation to a declared constant.
Parameters:
Y - 16-bit word result variable that is set to:
0 - if *X=L*
1 - if *X<L*
2 - if *X>L*
X - the input variable of an arbitrary type to be compared with *L;*
L - the value to which *X* is to be compared a decimal integer or floating-point number.

Section name - SIMULATION
Option name - **NEGATION**
Option value - *X,Y*
Meaning - executes logical negation of a variable.
Parameters:
X - 16-bit word result variable;
Y - 16-bit word input variable to be negated.

The correct operation of the module demands following changes in the definition of ASMEN's process variables:
- the redefinition of transmission channels used in the simulation on the NONE type,
- the change of conversion functions for simulated variables ONTO THE suitable versions of the NOTHING_xx functions.

EXAMPLE

```
[SIMULATION]
INIT=D1,10.18
INIT=D6,567
INIT=R2,1000
MOTOR=D2,D3,D4,D5,2048
```

RANDOM=D9,-10,10,2
RANDOM**=**D10,-15,20,1
REGULATOR=R1,R2,U1,21
REGULATOR=R1F,R2F,U2,11
RELATION=R3,-100,100,D5,2048
RELATION**=**R3F,0,4000,D5,2048

38. LIST OF FIGURES

Figure 2-1. Asix Help File - Contents Tab. 11
Figure 2-2. Asix Help File - Index Tab. 12
Figure 2-3. Asix Help File - Search Tab. 12
Figure 3-1. 'Welcome to the Asix Installation Wizard' Window. 17
Figure 3-2. 'License Agreement' Window. 17
Figure 3-3.'Readme Information' Window. 18
Figure 3-4. 'User Information' Window. 18
Figure 3-5. 'Destination Folder' Window. 19
Figure 3-6. Window Informing about asix Installation Performance. 21
Figure 3-7. 'Application Maintenance' Window. 22
Figure 3-8. Shortcut Properties Window. 23
Figure 4-1. 'Start Delay' Window. 31
Figure 5-1. 'User Interface Language of ASKOM Applications' Window -1. 33
Figure 5-2. 'User Interface Language of ASKOM Applications' Window -2. 34
Figure 5-3. 'User Interface Language of ASKOM Applications' Window -3. 34
Figure 5-4. 'User Interface Language of ASKOM Applications' Window - 4. 35
Figure 5-5. 'User Interface Language of ASKOM Applications' Window - 5. 35
Figure 5-6. 'User Interface Language of ASKOM Applications' Window - 6. 36
Figure 6-1.Adding new communication channel definition. 39
Figure 6-2. 'Writing data and status' parameter declaration. 41
Figure 6-3. Declaration of Network Channel – Obligatory Parameters. 74
Figure 6-4. Declaration of Network Channel - Optional Parameter. 75
Figure 6-5. GATEWAY Station Operating-Mode Declaration. 77
Figure 6-6. Declaration of Redundancy Channels. 78
Figure 6-7. Declaration of Remote Control Execution. 79
Figure 7-1. Configuration of Archival Data - Standard Data. 89
Figure 7-2. Configuration of Archival Data - 'Find Server Repeat Period' Parameter. 92
Figure 7-3. Configuration of Archival Data - 'Time for Server Profile' Parameter. 93
Figure 7-4. An Exemplary Declaration of Variables in VarDef. 95
Figure 7-5. Parameterization of Replenishment for B-Type Archive. 102
Figure 9-1. The Designer Window. 118
Figure 9-2. 'File Manager' Window. 120
Figure 9-3. 'Mask Text Description Generation' Window. 121
Figure 9-4. 'Mask Graphical Print' Window. 122
Figure 9-5. 'Mask Definition' Window. 124
Figure 9-6. 'Mask Selection' Window. 127
Figure 9-7. The 'NUMBER' Window for Process Variables Edition. 132
Figure 9-8. The 'NUMBER' Window for Coordinates Edition. 132
Figure 9-9. The Window for Cursor Snap Setting. 134
Figure 9-10. The 'Process Variables Exchange' Window. 137
Figure 9-11. The Window for Definition the Template of Grouped Objects. 138
Figure 9-12. 'Key Set Selection' Window. 142
Figure 9-13. The Window for Key Set Definition. 143
Figure 9-14. The Window for Assigning a Single Key to an Action. 144
Figure 9-15. The Window for Entering the Password to the asix System. 146
Figure 9-16. 'Password Manager' Window. 147
Figure 9-17. The Window for Password Change. 148
Figure 9-18. 'User Manager' Window. 149
Figure 9-19. The Window for asix's Users Definition. 150
Figure 9-20. The 'User Logon' Window. 150
Figure 9-21. The Window of Global Policies Manager. 153
Figure 9-22. The Window for the PICTURE Class Object Definition. 155
Figure 9-23. The Window for Bitmap Selection. 155
Figure 9-24. 'Bitmap Editor' Window. 157
Figure 9-25. The Window for New Bitmap or Icon Creation. 157
Figure 9-26. The Window for Edited Bitmap Color Definition. 159
Figure 9-27. The Expression Generator Window. 162
Figure 9-28. 'Items view' Window. 163
Figure 9-29. The Window for the TEXT Class Object Parameterization. 165
Figure 9-30. The Window for Font Handling. 166
Figure 9-31. 'Macro Manager' Window. 170
Figure 10-1 .Screen keyboard. 183
Figure 10-2. Numeric keyboard. 184
Figure 12-1. 'Alarms Exclusions' Window. 213
Figure 12-2. 'Exclude files' Window. 214
Figure 12-3. The Algorithm of Alarm Filtering (1). 215
Figure 12-4. The Algorithm of Alarm Filtering (2). 216

Figure 12-5. 'Active Alarms' Window. 220
Figure 12-6. 'Historical Alarms' Window. 220
Figure 12-7.'Selection Criteria' Window. 223
Figure 12-8. 'Alarm Selection Files' Window. 224
Figure 12-9. 'Alarms Exclusions' Window. 225
Figure 12-10. 'Exclude Files' Window. 226
Figure 12-11. 'Alarms Filtering' Window. 226
Figure 12-12. 'Alarms Signals' Window. 227
Figure 12-13. 'Historical Alarms Printout' Window. 228
Figure 12-14. 'Alarm Definition' Window. 229
Figure 12-15. 'Connection to Historical Alarms' Window. 229
Figure 12-16. 'Connection to Historical Alarms' Window - Advanced Option. 230
Figure 12-17. 'Historical Alarms Mask Definition' Window. 231
Figure 12-18. 'Selection Criteria' Window. 233
Figure 13-1. 'Reporter' Window. 236
Figure 13-2. The Window with Exemplary Report. 237
Figure 13-3. The Window with the Information on Deleted Report. 238
Figure 13-4. The Window for Report Directory Selection. 239
Figure 13-5. The Window of Status of Report Definition Correctness. 239
Figure 13-6. The Window of Report Definition Editor. 241
Figure 13-7. The Window for Inputting a Notice into a Report. 242
Figure 13-8. The Window for Date Setting of Report Generation. 242
Figure 13-9. The Menu of the Report Displaying Window. 243
Figure 13-10. The Graphical Presentation of Start and End of Exceeding. 272
Figure 17-1. 'Action Scheduler' Window. 336
Figure 17-2. The Window for Event Schedule Definition. 337
Figure 17-3. 'Action Execution by Scheduler' Window. 339
Figure 18-1. 'Blinking' Window. 345
Figure 18-2. The AsBase Object Parameterization Window. 348
Figure 18-3. The Exemplary Definition of Variable Set in the AsBase Module. 351
Figure 18-4. An Example of BAR. 352
Figure 18-5. The BAR Object Parameterization Window. 353
Figure 18-6. The BUTTON Example. 358
Figure 18-7. The BUTTON Object Parameterization Window. 358
Figure 18-8. The CALCULATOR Object Parameterization Window. 362
Figure 18-9. 'Expression Generator' Window. 364
Figure 18-10. The DATE+TIME Example. 366
Figure 18-11. The DATE + TIME Object Parameterization Window. 367
Figure 18-12. The ELIPSE Example. 369
Figure 18-13. The ELIPSE Object Parameterization Window. 370
Figure 18-14. The ELIPSES Object Parameterization Window. 371
Figure 18-15. The 'Coder' Window. 375
Figure 18-16. The EXPRESSION Object Parameterization Window. 378
Figure 18-17. The LINE and LINE HV Object Parameterization Window. 384
Figure 18-18. The LINES and LINES HV Object Parameterization Window. 386
Figure 18-19. The 'Coder' Window. 390
Figure 18-20. The MESSAGES Example. 392
Figure 18-21. The MESSAGES Object Parameterization Window. 393
Figure 18-22.The MOVEMENT CONTROLLER configuration window. 398
Figure 18-23. Examples of MOTOR type objects. 399
Figure 18-24. The MOTOR Object Parameterization Window. 399
Figure 18-25. The NUMEBR Example. 402
Figure 18-26. The NUMBER Object Parameterization Window. 403
Figure 18-27. The OBJECTS CONTROLLER Configuration Window. 411
Figure 18-28. The STRING Example. 412
Figure 18-29. The PICTURE Example. 415
Figure 18-30. The PICTURE Object Selection Window. 416
Figure 18-31. The PICTURES Example. 417
Figure 18-32. The PICTURES Object Parameterization Window. 417
Figure 18-33. The 'Coder' Window. 421
Figure 18-34. The PIPELINE Example. 423
Figure 18-35. The PIPELINE Object Parameterization Window. 424
Figure 18-36. The 'Coder' Window. 426
Figure 18-37. The PIONTER Examples. 428
Figure 18-38. The PIONTER Object Parameterization Window. 429
Figure 18-39. The POLYGON Examples. 434
Figure 18-40. The POLYGON Object Parameterization Window. 434
Figure 18-41. The POLYGONS Object Parameterization Window. 436
Figure 18-42. The 'Coder' Window. 439
Figure 18-43. The POLYLINE Object Parameterization Window. 442
Figure 18-44. The POLYLINES Object Parameterization Window. 444

Figure 18-45. The 'Coder' Window. ... 447
Figure 18-46. The PRESENTER Example. ... 450
Figure 18-47. The PRESENTER Object Parameterization Window. ... 451
Figure 18-48. The RECTANGLE Examples. ... 453
Figure 18-49. The RECTANGLE Object Parameterization Window. ... 454
Figure 18-50. The RECTANGLES Object Parameterization Window. ... 456
Figure 18-51. The 'Coder' Window. ... 460
Figure 18-52. The REFRESH TEST Object Parameterization Window. ... 462
Figure 18-53. The REPORT Example. ... 463
Figure 18-54. The REPORT Object Parameterization Window. ... 465
Figure 18-55. The SELECTOR Examples. ... 467
Figure 18-56. The SELECTOR Object Parameters Window. ... 468
Figure 18-57. The SLIDER Examples. ... 474
Figure 18-58. The SLIDER Object Parameterization Window. ... 475
Figure 18-59. The SWITCH Examples. ... 479
Figure 18-60. The 'Control' Window for Action Definition for the Button. ... 479
Figure 18-61. The SWITCH Object Parameterization Window. ... 480
Figure 18-62. The SWITCH SET Example. ... 487
Figure 18-63. The SWITCH SET Object Parameters Window. ... 487
Figure 18-64. The SYNCHRONIZER Object Parameters Window. ... 491
Figure 18-65. The TEXT Object Parameters Window. ... 493
Figure 18-66. The TEXTS Examples. ... 496
Figure 18-67. The TEXTS Object Parameters Window. ... 497
Figure 18-68. The 'Coder' Window. ... 498
Figure 18-69.Declaration of Texts for Monitored Variable States in Variable Definictions Database. ... 504
Figure 18-70. The WORK POINT Object Parameters Window. ... 505
Figure 18-71. The CHART Object Example. ... 508
Figure 18-72. The Context-Sensitive Menu of the CHART Object. ... 511
Figure 18-73. The CHART Toolbar. ... 512
Figure 18-74. The CHART Toolbar. ... 512
Figure 18-75. The 'CHART Functions' Window - Cursor Tab. ... 516
Figure 18-76. The 'CHART Functions' Window - Chart Service Tab. ... 516
Figure 18-77. The Chart Legend Window. ... 517
Figure 18-78. The 'Change Curve Definition' Window. ... 517
Figure 18-79. The 'CHART Functions' Window - Axis Zoom Tab. ... 517
Figure 18-80. The 'Time Horizons' Window. ... 518
Figure 18-81. The 'CHART Functions' Window - Archive Tab. ... 519
Figure 18-82. The 'Begin of Archive' Window. ... 519
Figure 18-83. The CHART Parameters Window. ... 521
Figure 18-84. The 'Chart Colors' Window. ... 523
Figure 18-85. The 'Cursor Parameters' Window. ... 524
Figure 18-86. The 'Curve Manager' Window. ... 524
Figure 18-87. The 'Curve Parameters' Window. ... 525
Figure 18-88. The 'CHART Parameters' Window. ... 528
Figure 18-89. The 'Curve Parameters' Window. ... 529
Figure 21-1. The Translation File Generation Window. ... 542
Figure 31-1. The Channel Manager Main Window. ... 561
Figure 31-2. The Channel Manager Window. ... 562
Figure 31-3. The Channel Options Window. ... 562
Figure 31-4. The 'Channel OPC' Window. ... 563

39. LIST OF TABLES

Table 8-1. Logical Adapters of Aslink. 113
Table 12-1. OPC Alarm Strategy - the Table for the Example. 207
Table 13-1. The List of Operators and Functors for the ASTEL Language. 278
Table 13-2. Examples of Expressions for the ASTEL Language. 279
Table 13-3. The List of Error Types Typical For Report Generation. 285
Table 13-4. The list of Parameters Transferred to Scripts that Generate Reports. 307
Table 18-1. Identifiers of Constant Fields of AsBase Database Tables. 350
Table 18-2. Functions with Suitable Key Code and Graphic Representation for the CHART Object. 513
Table 18-3. Functions with Suitable Key Code and Graphic Representation for the CHART Object (continuation). 513
Table 18-4. Functions with Suitable Key Code and Graphic Representation for the CHART Object (continuation). 514
Table 18-5. Functions with Suitable Key Code and Graphic Representation for the CHART Object (continuation). 514